



PROCESS FLOW FEATURES AS A HOST-BASED EVENT KNOWLEDGE  
REPRESENTATION

THESIS

Benhur E. Pacer Jr, Captain USAF

AFIT/GCS/ENG/12-06

DEPARTMENT OF THE AIR FORCE  
AIR UNIVERSITY

**AIR FORCE INSTITUTE OF TECHNOLOGY**

Wright-Patterson Air Force Base, Ohio

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the United States Government. This material is declared a work of the U.S Government and is not subject to copyright protection in the United States.

AFIT/GCS/ENG/12-06

PROCESS FLOW FEATURES AS A HOST-BASED EVENT  
KNOWLEDGE REPRESENTATION

THESIS

Presented to the Faculty  
Department of Electrical and Computer Engineering  
Graduate School of Engineering and Management  
Air Force Institute of Technology  
Air University  
Air Education and Training Command  
In Partial Fulfillment of the Requirements for the  
Degree of Master of Science

Benhur E. Pacer Jr, B.S. Computer Engineering  
Captain USAF

June 2012

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AFIT/GCS/ENG/12-06

PROCESS FLOW FEATURES AS A HOST-BASED EVENT  
KNOWLEDGE REPRESENTATION

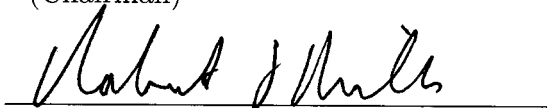
Benhur E. Pacer Jr, B.S. Computer Engineering  
Captain USAF

Approved:



Dr. Gilbert L. Peterson  
(Chairman)

31 MAY 2012  
date



Dr. Robert F. Mills  
(Member)

1 JUN 2012  
date



Dr. Michael R. Grimaila, CISM, CISSP  
(Member)

31 MAY 2012  
date

*Abstract*

Malicious software, or malware, are programs with malicious intent [1], and can exist in the form of viruses, worms, trojan horses, spyware, and rootkits. The detection of malware is of great importance but even non-malicious software can be used for malicious purposes. Monitoring processes and their associated information can characterize normal behavior and help identify malicious processes or malicious use of normal process by measuring deviations from the learned baseline. This exploratory research describes a novel host feature generation process that calculates statistics of an executing process during a window of time called a process flow. Process flows are calculated from key process data structures extracted from computer memory using virtual machine introspection. Each flow cluster generated using  $k$ -means of the flow features represents a behavior where the members of the cluster all exhibit similar behavior. Testing explores associations between behavior and process flows that in the future may be useful for detecting unauthorized behavior or behavioral trends on a host. Analysis of two data collections demonstrate that this novel way of thinking of process behavior as process flows can produce baseline models in the form of clusters that do represent specific behaviors.

## *Acknowledgements*

To my advisor Dr. Gilbert Peterson, thank you for your help and getting me past some rough times. I had many technical road blocks, thank you for calming me down when the stress level was high and providing the much needed mentorship.

To my parents, thank you for all your support. You both were always there when I needed someone to talk to when facing burnout from school and kept me motivated.

Benhur E. Pacer Jr

# *Table of Contents*

	Page
Abstract . . . . .	iv
Acknowledgements . . . . .	v
List of Figures . . . . .	ix
List of Tables . . . . .	x
List of Abbreviations . . . . .	xii
 I. Introduction . . . . .	 1
1.1 Impact of Research . . . . .	2
1.2 Research Overview . . . . .	2
1.2.1 Problem Domain. . . . .	2
1.2.2 Past Research . . . . .	3
1.2.3 Problem Statement and Hypothesis . . . . .	3
1.2.4 Data Source . . . . .	4
1.2.5 Assumptions and Limitations . . . . .	5
1.2.6 Results . . . . .	6
1.3 Summary . . . . .	6
 II. Literature Review . . . . .	 8
2.1 Intrusion and Computer Security . . . . .	8
2.2 Evidence of an intrusion . . . . .	9
2.3 Data collection implementation . . . . .	9
2.4 Intrusion Detection Systems (IDS) . . . . .	10
2.4.1 Approach Methodologies . . . . .	12
2.4.2 Statistical and Machine Learning Techniques For Intrusion Detection . . . . .	18
2.4.3 Network based intrusion detection systems (NIDS)	24
2.4.4 Host based intrusion detection systems (HIDS) .	27
2.5 Flow-Based Features . . . . .	33
2.5.1 Network Based Flow-Based Features . . . . .	34
2.6 Summary . . . . .	35

	Page
III. Methodology . . . . .	36
3.1 Problem Definition . . . . .	36
3.1.1 Research Goals . . . . .	37
3.1.2 Expected Outcome . . . . .	38
3.1.3 Assumptions . . . . .	38
3.2 Research Approach . . . . .	39
3.3 Host Based Process Flow Features . . . . .	39
3.3.1 Compiled Memory Analysis Tool (CMAT) . . . . .	40
3.3.2 Data Preparation . . . . .	41
3.3.3 Host-Based Process Flow Feature Generation . . . . .	43
3.3.4 Process Flow and Features . . . . .	44
3.4 Clustering . . . . .	46
3.5 Similarity Measure . . . . .	48
3.6 Designator Analysis of Host-Based Process Flow Feature . . . . .	48
3.6.1 Script Data, Controlled Experiments . . . . .	48
3.6.2 Uncontrolled Experiments . . . . .	50
3.7 Disadvantage of the Selected Research Methodology . . . . .	52
3.8 Summary . . . . .	53
IV. Results Analysis . . . . .	55
4.1 Controlled Experiment . . . . .	55
4.1.1 Parameter Settings . . . . .	56
4.1.2 Data Efficacy of Clusters on Individual Processes . . . . .	60
4.1.3 Tracking distinct processes . . . . .	62
4.1.4 Test of Clusters . . . . .	68
4.1.5 Data Collect and Problems . . . . .	70
4.2 Unlabeled Data Collect: Uncontrolled Experiment . . . . .	72
4.2.1 Window 2003 OS, CMAT Captures . . . . .	72
4.3 Windows7 Operating System HACKFEST . . . . .	74
4.3.1 Cluster Analysis For Windows 7 OS: BSOD - 19 . . . . .	74
4.3.2 BSOD - 19 Amount of Clusters needed . . . . .	82
4.3.3 Using BSOD - 19 clusters with other Data Set (CAE-18) . . . . .	84
4.3.4 Data Collect and Problems . . . . .	87
4.4 Summary . . . . .	88



	Page
V. Conclusions . . . . .	90
5.1 Limitations and Assumptions . . . . .	91
5.1.1 Sensor Impact and Virtual Environment . . . . .	92
5.1.2 Memory as an Observable Environment . . . . .	92
5.2 Contributions . . . . .	92
5.3 Future Work . . . . .	93
Appendix A. Normal Activity Script . . . . .	95
A.1 Script Labeled Data Collect . . . . .	95
Bibliography . . . . .	100

## *List of Figures*

Figure		Page
2.1.	Teodoro et al., The components an IDSs from the definition from the CIDE [2] . . . . .	12
2.2.	Mukhopadhyay et al., The components of Intrusion Detection and Prevention System for a signature system [3] . . . . .	13
2.3.	Laureano et al. HawkEye Flow Diagram [3] . . . . .	14
2.4.	Shon and Moon Enhanced SVM approach to an IDS [4] . . . . .	21
2.5.	Laureano, et al. Model for protecting HIDS sensors using virtual machines [5]. . . . .	29
2.6.	Malan et, al. Model HIDS for detection of worms [6]. . . . .	30
2.7.	Feng, et al. HIDS model with alert fusion [7]. . . . .	31
4.1.	Index Score for Window 5. . . . .	59
4.2.	Index Score for Window 10. . . . .	60
4.3.	DLL Features for Cluster 15. . . . .	64
4.4.	Average DLL and File Handle Permissions Features for Cluster 15. . . . .	65
4.5.	DLL Features for Cluster 89. . . . .	66
4.6.	Average DLL and File Handle Permissions Features for Cluster 89. . . . .	67
4.7.	DLL Features for Cluster 98. . . . .	68
4.8.	Average DLL and File Handle Permissions Features for Cluster 98. . . . .	69
4.9.	Davies-Boldin Dunn Index Sliding Window 5 on Windows 7 . . . . .	82
4.10.	Davies-Boldin Dunn Index Sliding Window 10 on Windows 7 . . . . .	83
4.11.	Davies-Boldin Dunn Index Sliding Window 20 on Windows 7 . . . . .	83

## *List of Tables*

Table	Page
3.1. The Database Tables Created from the CMAT Feature Files Output . . . . .	42
3.2. Raw CMAT Output Files . . . . .	42
3.3. Process Flow Characteristics . . . . .	44
3.4. An instance of Processes Flow and Features . . . . .	45
4.1. Timeline for Controlled Experiment. . . . .	55
4.2. Computer 7_4 $k - Means3resultsofthemostpopulatedclusters$ . . . . .	61
4.3. SearchProtocol and Accompanying Processes. . . . .	61
4.4. Controlled Experiments Process Flows. . . . .	62
4.5. Controlled Experiments Process Flows with DLL Features. . . . .	63
4.6. Controlled Experiments Clusters File Handles with Permissions. . . . .	63
4.7. Controlled Experiments Process Flows Registry Key Features. . . . .	63
4.8. Timeline for Controlled Experiments with Anomalous Processes. . . . .	70
4.9. The anomalous processes hiding under notepad.exe . . . . .	71
4.10. Hackfest BSOD-02 TimeLine. . . . .	73
4.11. Assigned PIDs Application Name. . . . .	73
4.12. Cluster Characteristics model for 16 clusters. . . . .	74
4.13. Hackfest BSOD-19 TimeLine . . . . .	74
4.14. Hackfest BSOD-19 Applications. . . . .	76
4.15. Hackfest BSOD-19 chrome.exe. . . . .	76
4.16. Hackfest BSOD-19 Initialization, Ports. . . . .	77
4.17. Hackfest BSOD-19 chrome.exe, iexplore.exe and nessus. . . . .	78
4.18. Hackfest BSOD-19 chrome.exe, iexplore.exe . . . . .	79
4.19. Hackfest BSOD-19 chrome.exe, iexplore.exe initialization cluster. . . . .	80
4.20. Hackfest BSOD-19 chrome.exe, iexplore.exe clusters. . . . .	81

Table		Page
4.21.	Hackfest BSOD-19 chrome.exe, iexplore.exe clusters. . . . .	81
4.22.	Hackfest CAE-18 TimeLine. . . . .	84
4.23.	Hackfest CAE-18 chrome.exe, iexplore.exe clusters . . . . .	84
A.1.	The normal activity script for computer . . . . .	95

## *List of Abbreviations*

Abbreviation		Page
DoD	Department of Defense . . . . .	1
IDS	Intrusion Detections Systems . . . . .	1
NIDS	Network-Based Intrusion Detection System . . . . .	2
HIDS	Host-Based Intrusion Detection System . . . . .	2
VMI	virtual machine introspection . . . . .	4
CMAT	Compiled Memory Analysis Tool . . . . .	4
GA	genetic algorithm . . . . .	22
VMI	virtual machine introspection . . . . .	36
OS	Operating System . . . . .	36
GUI	graphical user interface . . . . .	37
GUI	Graphical User Interface . . . . .	39
SCADA	supervisory control and data acquisition . . . . .	93
USAF	United States Air Force . . . . .	93

# PROCESS FLOW FEATURES AS A HOST-BASED EVENT KNOWLEDGE REPRESENTATION

## I. Introduction

The meaning of “cyberspace” is highly debated among United States military officials and the computer security community. The Department of Defense (DoD) has defined cyberspace as a war fighting domain [8]. A domain that is no different from air, land, sea and space where military operations and tactics can be used against advisories. Organizations and government agencies are in a constant race to develop new methods and tools to defend their computer assets from “attackers” from other nation states, terrorist organizations, or individuals. The development of intrusion detections systems (IDS) were developed to combat these threats of intrusions. IDS have two distinct strategies, using known signatures to identify attackers (signature-based IDS) or using baseline models that represent a range of normal states of a computer system (anomaly-based IDS).

Anomaly-based IDS has the advantage of detecting previously unseen novel attacks [9]. The tradeoff for an anomaly based system is it increases the number of false positive alerts [9]. Research on anomaly based IDS shows promising results but lacks the accuracy to implement in the commercial market. Some of the reasons for the lack of accuracy is the type of features that the system monitors. The data source should provide the most accurate and up to date information of the state of the computer. The sensor that monitors the data source has to be reliable as well as the sensor itself has to be protected from attacks. The most important aspect of anomaly based IDS is the training data must be diversified enough to capture most of the normal behavior of a computer system. The purpose of this research is to develop host computer memory understanding, and determine if a method could be used to effectively group similar processes together based on their behavior to focus a computer security operator to a potential threatening program on the system.

## ***1.1 Impact of Research***

A signature-based IDS is accurate in detecting known intrusions where the signature patterns are known [10]. New attack signatures are added to a scanning database when discovered, but before this update occurs an attacker could go undetected compromising target computer systems for months. With the growth of the internet and networked systems the updating of intrusion signatures would become man power intensive because a new entry to the scanning database may need to be made for every variant of the same attack.

For anomaly detection in computers systems, the decision space is too large to effectively find the best set of features to use for a Network-Based Intrusion Detection System (NIDS) and Host-Based Intrusion Detection System (HIDS) [11]. A process is the fundamental element of a program in memory. If the features of a process can be monitored a method can be developed to baseline that behavior. The “notepad.exe” process in one computer should have the same characteristics running in another computer. If a process identifies itself as “notepad.exe” but deviates from the known baseline behavior then an alert is made to a system administrator. In addition, processes with similar behavior should also have similar characteristics. This research identifies the data in memory associated with a process that characterizes its dynamic behavior. A model of a process characterizes the process behavior in memory and a method is developed to detect processes that deviates away from a given baseline.

## ***1.2 Research Overview***

*1.2.1 Problem Domain.* The cyberspace domain is an infinitely large search space and selecting the best subset of features that describes the domain is an intractable problem [12]. Situational awareness tools only observe a portion of the threat space. The limitation in storage space, processing power and time are factors a system must handle because it’s not possible to observe all combinations of threat in cyberspace. The tradeoff for all derived implementations of situational awareness tools are effectiveness or efficiency.

*1.2.2 Past Research.* Most of today’s current threat detection systems are signature based tools; to which updates are made as new threats are discovered [10]. The period of time when these threat are discovered and when updates are received can be long enough for threats such as malicious software to infect many computers. The attractiveness of anomaly based tools for detecting novel attacks has led to research on several methods to perform threat detection and classification [13] [14] [15] [2]. Chapter II presents an overview on some of the academic research, challenges and implementations of observing the threat space. Several research areas include using network traffic, host based features or a combination of both to determine intrusions [9] [16] [14] [17] [2] [18].

Eskin, et al. [19] developed a framework for unsupervised learning for anomaly detection with no reliance on labeled data. More specifically they proposed a clustering algorithm that uses a radius parameter assigned to clusters. The clustering algorithm then has the ability to detect outliers outside the threshold length of the radius. No research effort has been attempted to implement a clustering algorithm using the dynamic process behavior in memory.

In [20], Windows sysinternal suite of tools used in the digital forensics field for situational awareness was used as sensors for monitoring relevant operating system objects. These tools are a set of many little software programs that monitor only one type of feature or object in the operating system. The combination of all of these tools outputs was used to train an artificial neural network for detection of anomalies. In [21], network features that characterize statistical packet behavior called “flows” were combined with Windows sysinternal tools. The research combined both network and host information for a method to potentially detect intrusions.

*1.2.3 Problem Statement and Hypothesis.* A “process” is a program in execution. The difference between a program and processes is a program is a passive entity in a computer and a process is an active entity associated with resources. As programs execute in memory the accessing of memory features provides a great data



source for cyberspace situational awareness. According to Carvey [22], some malware may only exist in memory evading anti-virus signature scans.

From the data in memory an abstraction to features of processes called flows are made. Clustering of the flows groups similarly behaving process together. These groups provide the correlation to individual processes with its behavior. Grouping known processes together provides the means for finding potential threats. The hypothesis for this research is that these groupings exists when using process flow features extracted from memory using virtual machine introspection.

The goal of this research is to use virtual machine introspection(VMI)to evaluate the novel use of host based process flow feature clustering to model processes behaviors. The system is categorized as Host-Based detection with the data of interest as memory features. The Compiled Memory Analysis Tool (CMAT) is a forensic analysis tool that provides the VMI sensor, capable of live memory captures of a system. It works by capturing the host operating system memory while operating in a hyper visor environment, producing five feature files approximately every 30 seconds and one file that relates to registry key information every 30 minutes.

*1.2.4 Data Source.* Two data sources were used in the training and testing. The first data set consists of a scripted event to provide time stamps for when applications were running and provides a means to generate a labeled data set. This test focuses on encapsulating different computer applications of the same type, for example three different internet browser were used. A known anomalous process was created using a free software tool that masquerades as a different processes. The masquerading processes and the other test data was manually labeled. Testing of the resultant behavior clusters shows that similar process behaviors do group together.

The second data set was collected from a set of computers in an isolated network during the ACE HACKFEST exercise. The HACKFEST exercise consisted of two teams that had the objective to defend there assigned computer assets and at the same time attack the other teams assets. The CMAT sensor was placed on a Xen

hypervisor monitoring Window7 and Windows 2003 guest operating systems. Out of the 20 Windows 7 and Windows 2003 computer, less than half were usable due to data constraints. It was discovered that some of the CMAT sensors crashed at some point and was not discovered until the exercise was over, resulting in incomplete or no information. Another problem with the CMAT sensor outputs was the frequency of the sensor outputs was inconsistent. This was an important aspect in characterizing processes behavior in a given time window, those outputs that had very dissimilar frequencies were eliminated as a data source. In the standard configuration of CMAT, five feature files output every 30 seconds and a memory dump with another feature file is captured every 30 minutes. An artifact of this is when a memory dump is captured the five output feature files are delayed for more that a couple of minutes. In spite of this problem, using the acceptable data shows that chrome.exe and iexplore.exe cluster initialization phase identifiable different process behave the same.

*1.2.5 Assumptions and Limitations.* The limitations of CMAT are that the sensor feature file outputs were inconsistent mostly because of a software crash during the exercise with nobody taking notice to restart the sensor. The HACKFEST exercise was an uncontrolled event, but looking at the activity logs most of the attacks did not inject or spawn anomalous processes in the target system. The attacks were denial of service types that the information in memory would not reflect as anomalous. The big assumption was that the data set used for training was considered clean from anomalous processes. Because the CMAT sensor feature files outputs are delayed more than a few minute when the memory dump is captured every 30 minutes a noticeable gap in time was observed. Between these events, the file outputs every 17 to 65 seconds. These temporal inconsistencies are dropped when extracting the feature file over a given time window. In other words, the file outputs are considered equal separation. If the time window that describes the behavior of a processes is 10,  $\{T_0, T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_8, T_9, T_{10}\}$  an assumption is that the time lapsed between

each discrete time is the same. A process flow is then derived from this sliding window of 10 discrete time instances.

*1.2.6 Results.* The results of the experiments provided an initial indication that behavioral relationship can be correlated to clusters. Activities on a computer for two data sets were generated and a correlation between clusters and process behavior were observed. This research was exploratory and the goal was to examine if a connections of process behavior to the statistical feature set calculated from processes executing in memory was possible. No definitive conclusions were made but a strong trend was observed that clustering process flows did provide distinct behavior relationships. An example is a cluster with network work activity included applications such as internet browsers. The cluster would be labeled a communication cluster and an application like an internet browser would be assigned to this cluster when the behavior of communicating over the internet is observed. A fraction of the collected data set was evaluated based on isolating valid data from invalid data. The valid data had consistent frequencies of CMAT outputs. With this small data sets four processes Outlook, notepad.exe, calc.exe, and explorer.exe were used to show a trend that similar processes in different computers behaved similarly and similar processes in the same computer behaved similarly as they were assigned to the same clusters.

### **1.3 Summary**

The clustering algorithm successfully grouped liked process applications together exploratory observations was made by looking at the application names as it presents itself in the CMAT output files. New instances of a processes flows were successfully identified by application types using the Euclidian distance measure from the training cluster's centroid. The training data must be diversified, clean and accurately describes a standard set of processes in a computer. A computer in a military network would have different set of applications running than a home computer because of the purpose of the computer system. The 194 feature set implemented in

this research could be used in future research for modeling different types of computer systems with distinct different purposes. This method can be implemented on a system that has a specific task.

## II. Literature Review

This chapter presents an overview of prior research and existing technologies to perform attack classification using relevant host memory features. This chapter provides background of the current research and approaches using purely host features, network features or a combination of both. The topics include a discussion of intrusion detection system followed by an overview of most machine learning techniques used for detecting intrusions.

### *2.1 Intrusion and Computer Security*

Intrusion detection systems are increasing in importance because of the greater role computers play in everyday life. Cyber-criminals and hackers are using this increase dependency on computers to target more individuals for intrusion. Today IDSs play a vital role in combination with intrusion prevention methods such as authentication, secure software design, and information protection like encryption. Easy accessibility to the internet and networks has increased the amount of potential threats from hackers [3].

Intrusions are events or actions that attempts to compromise the confidentiality, integrity, or availability of resources [10]. Intrusion detection is the process of monitoring and collecting data from the target computer. The gathered data is analyzed to find evidence of an intrusion. Intrusion detection system (IDS) is the automated process that implements intrusion detection [10] [13].

A variety of intrusions exists from programs that damage computer systems such as worms, viruses and trojan horses to theft related intrusions with the purpose on stealing information such as computer theft, financial fraud and military and business secrets. A complete prevention of intrusions is not possible so detecting intrusions is an important aspect of security to minimize the damage as a result of intrusions [23].

## ***2.2 Evidence of an intrusion***

A computer is a system that stores data and processes data. Thinking of the data in a computer as evidence when investigating an intrusion, the practice of forensic science can be applied to answer the fundamental questions of “What”, ”Where”, “When”, “How”, and “Who” relating to the attack or intrusion. The challenge is to apply forensic science to a computer system with a large amount of data and do the analysis fast to have a real time system. According to Inman and Rudden [24], forensic science is an applied science based on the laws of physics and chemistry. The five concepts of evidences are transfer, identification, individualization, association, and reconstruction. Inman and Rudden proposes a sixth concept, the idea of matter dividing before transfer. The initial key step is identifying the evidence to analyze. In designing an IDSs the identifying of evidence to analyze is very difficult because of the wide range of computer models and the amount of data the computer can hold.

According to Inman and Rudden matter must divide before it can be transferred. In the digital world and dealing with digital evidence we can take this concept and for components of the divided digital evidence it holds that these component parts will have characteristics from the original components. Digital evidence from an attack or intrusion will have characteristics of an attack or intrusion. Determining these characteristics is very difficult, instead we can take the evidence in a computer system performing and behaving normal then gather evidence. From the gathered evidence we can define characteristics as normal or not an intrusion or attack. An IDS will monitor the same evidence and if it deviates in characteristics from normal it is assumed to be an intrusion and an alarm is set.

## ***2.3 Data collection implementation***

There are two main methods to collect data, interval-based or continuous [10]. The interval-based data collect method also referred to batch mode collects the data in a non-continues format. This method is usually found in HIDS where the data are the log events generated in a file. The continuous method uses a continuous flow of

information and is usually implemented in NIDS where the flow of information are the packets of the network [10].

An IDS is only as good as the data it collects. The data must be reliable and complete [13]. The Windows operating system provides an auditing mechanism with several logs available for use. The most commonly used logs are the application, security and system logs. The application log event entries are triggered with applications running on the host system. The security log contains events of login information and events related to resources on the system. The system logs contains system component information such as driver failure and hardware issues. Some logs are customizable for use by the IDS, with some auditing system may be configured to record every single system calls invoked by every process in the operating system [13].

Collecting data without analysis is useless [13]. Auditing a computer system has no benefits if the data collected is not used. There are two strategies on how to analyze the data: misuse detection method and anomaly detection method.

## ***2.4 Intrusion Detection Systems (IDS)***

An IDS is a system that detects evidence of an intrusion [9] [13]. The system initiates responses depending on the implementation. The most common response is a warning to the system administrator or user such as a pop up window with a description of the intrusion. The purpose of an IDS is to collect data on the computer system it monitors and initiate some action in response after detecting evidence of an intrusion [13]. A manual process was the predecessor and an automated process was developed because it overwhelmed the system administrator whose task was to find intrusion form the data collected.

A current computer, commercial of the shelf, comes with a firewall, an operating system with automated patching if connected to the internet, virus protection and has options to password protect resources. With the current security architecture in place, it is necessary to add an IDS. The IDS gives an extra protection against

attacks. With any security architecture in place an intrusion may still occur. [13] An IDS does not stop all attacker from penetrating the computer security but will make it more difficult. It acts as a deterrent from intruders attempting to attack if the known system has an IDS installed. According to Bace et al [10], the goal of the computer security management team is to affect the behavior of users. If there is a risk of getting caught doing some illegal activity the users are less likely to commit a crime compromising the confidentiality, integrity and availability of resources stored in organization's computer system.

According to [2], there are four functional modules in an IDS: An event block that acquires information called events, a database block that stores information from the event block, an analysis block that analyzes the data from the database block, and the response block that takes action against potential threats to the information system. An IDS is categorized by the type of data it collects (network data or host data), the analysis methodology (misuse or anomaly detection) and actions with response to an indication by the IDS (active or passive) [10]. A signature based method requires a database of known signatures and is updated when new attacks are found. Anomaly based detection does not require a database of known signatures instead uses a model of an estimate defining normal behavior of the system. Most systems uses a model of normal behavior but another method described in [2], can use a model of abnormal behavior. An alarm is generated when the given instance of an observation is within the threshold of the abnormal behavior model. The ultimate goal of an IDS is to detect all intrusions (100% detection rate) with no false alarms (0% false positive indication) but is very difficult, almost impossible. The set of features or data selected for the analysis is important to minimizing the workload of the host system. When the set of features is as small as possible the IDS is most efficient with respect to speed. An IDS has many underlying optimization problems associated with it. An IDS developer wants to maximize detection rate, minimize false positives indication, maximize speed of computation, and minimize features selected all at the same time. This multiple step optimization problem is an NP-hard problem. [25].



A working group created by DARPA in 1998 with the purpose of defining a common framework in IDSs called the “Common Intrusion Detection Framework” then later called the “Intrusion Detection Working Group” defined a general architecture based on four functional components [2]. Figure 2.1 shows the model from the working group developed. The E-boxes are described as the sensor that gathers the raw from the observed environment. The D-boxes are the database components that stores the data from the E blocks. The A-boxes are the analysis portions of the model and it analyzes the data from the database. The R block is the reactionary block of the model if the system has the ability to apply a reaction to an intrusion.

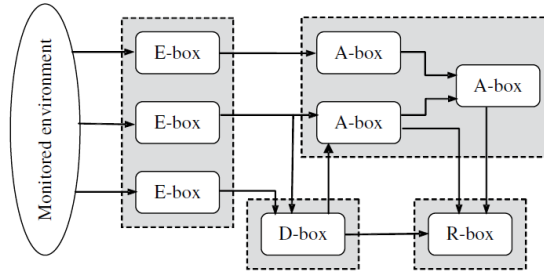


Figure 2.1: Teodoro et al., The components an IDSs from the definition from the CIDF [2]

*2.4.1 Approach Methodologies.* We have all this data from the IDS, network or host data, can we use this to detect intrusions in the future? The two analysis methods for detecting intrusions are misuse and anomaly detection [10] [7]. According to Bace et al. misuse detection is finding what is known, in this case the intrusion itself. Anomaly detection is finding how different a behavior characterized with values in a computer system from a model of normal activity. If any differences with respect to a threshold value then the snapshot of the computer that characterizes its behavior is labeled as an intrusion.

*2.4.1.1 Misuse detection using signatures (Knowledge Based).* The IDS with a signature based methodology uses known models of an attack. The known model is stored in a knowledge database containing every attack the system can

detect. A signature methodology can only detect an intrusion that has knowledge of by inserting the signatures into the database manually or through software updates via downloading from the vendor's website [10]

Current IDS products mainly use misuse analysis methodology to detect intrusion [10]. The system looks for pattern and these patterns are referred to as signatures of an attack. The system monitors either network data, host data, or both and with this data a search and compare is performed to match a predefined event or sets of events has occurs [10]. The events or set of events is the signature of an attack. Every attack with the signatures known by the system can detect the intrusion. Any new attack without a signature defined will go undetected.

The main advantage of misuse detection is the accuracy the system provides [10]. An IDS with misuse detection generates very low false positives because the system compares and looks for known attacks with exact known signatures. There is a high level of confidence that what the system labels as an attack is really an attack. In many academic literature the disadvantage of misuse detection is the inability to detect novel intrusions [10] [14].

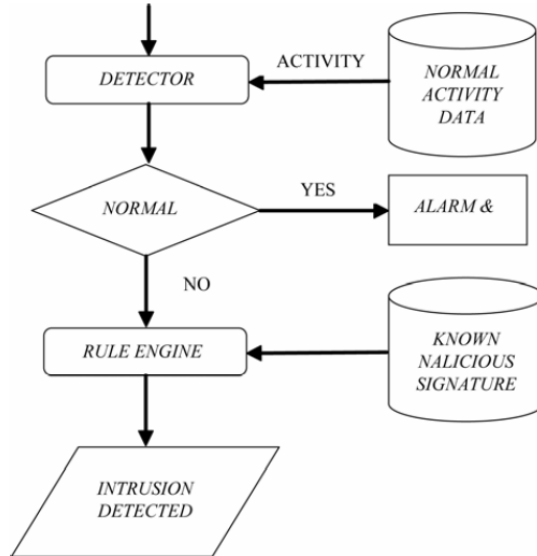


Figure 2.2: Mukhopadhyay et al., The components of Intrusion Detection and Prevention System for a signature system [3]

In [3] Laureano et. al proposed a method that uses a signature based intrusion detection and prevention system called HawkEye. Its a system that contains 5 specialized components: sensor, management server, database, console, and a demilitarized zone. The basic system flow is shown in figure 2.3. An event record is created then written to a file. The target agent or program then sends the file to the command console. The detection engine is used to match signatures from the recorded events of the file. A log system records all raw data from the incoming recorded events from the target system. If a known signature is found from then an alarm is generated. A response is generated to the target system after an alarm is generated and it gives system the ability to react to a threat.

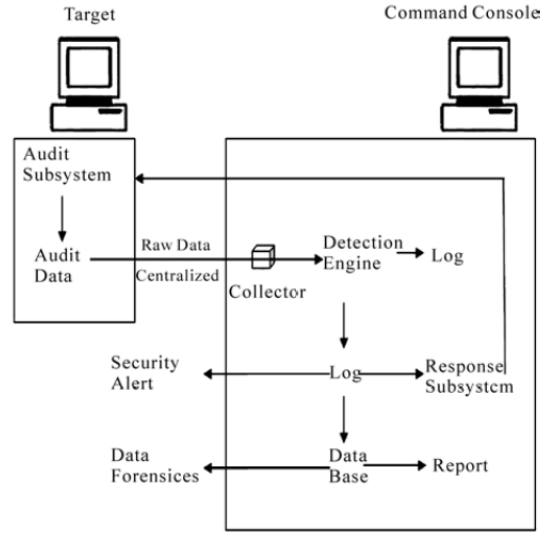


Figure 2.3: Laureano et al. HawkEye Flow Diagram [3]

*2.4.1.2 Anomaly Detection (Behavioral Based).* Traditional intrusion detection system based on signature matching or pattern matching is outdated because of the sophistication of the types of attacks used [26]. Anomaly detection with the ability to detect novel attacks has lead to numerous research to improve the accuracy and reduce the false positives, that past systems had problems with [7] [23]. A model of normal activity is compared to the gathered data of the target computer system. If the data deviates from the model, the state when the data was collected

is flagged as an intrusive activity [13] [23] [2]. Anomaly detection assumes everything that deviates from the normal activity model is intrusive. The comparison of data to the model is the easy part, the difficult task is coming up with the model that represents normal activity. Anomaly detection needs relevant data to train the IDS. The model constructed is from historical data collected over a period of time called the gathering and learning phase of the system.

Anomaly detection is not meant to replace an existing systems using a misuse detection method but is better implemented as an add-on system [14]. The earliest work using anomaly detection was by Dorothy Denning. Denning used a host-level system that modeled user profiles activity. Some of the features the system monitored were login frequency, password attempts/failures, session duration and computer resource consumption. The assumption was attacks or intrusions were different in terms of the monitored data from normal computer behavior or normal network traffic.

The Anomaly based detection is very appealing because of the ability to find intrusion without any prior knowledge of the attack or intrusive characteristics. New threats change constantly and new methods must be developed to detect these threats. In recent years anomaly based intrusion detection systems either using network or host data are just starting to appear in practice [2]. Anomaly based IDS's lack of presence in network or host security systems are because the numerous false positive alerts current systems are plagued with.

The polymorphic behavior of modern hackers is making signature based IDS almost impossible to catch. A hacker can apply different variations to an attack used to a vulnerability. These little variations would have to correspond to a different signature in an expert system IDS in order for the IDS system to detect every variation of the same attack [26]. Anomaly detection is the natural go-to method to counter the problem of having to generate every signature for every attack. Instead of many models that represent every attack called an attack signature, the anomaly detection method attempts to model normal activity in a computer system in one model. Any

deviation away from this one model is then considered anomalous. In this system all anomalous activity is assumed to be an intrusion and an alarm is set [26].

Threshold detection, statistical measures, rule based measures, neural networks, genetic algorithms and immune system are some of the methods used for calculating the deviations from the collected data that represents the computer's behavior to the normal behavior model [10] [23]. Threshold detection method monitors certain attributes of a user's behavior that is countable or measurable and if the monitored attribute exceeds the threshold defined by the developers of the IDS then an alarm indicating an intrusion has occurred.

Statistical measures are another method used for analyzing data for detecting intrusions. This assumes the distribution of the attributes of the profile describing the normal behavior could be modeled in traditional well known statistical models [10] [23]. If the distribution is unknown then a learning step is required to learn the distribution from past observed data of the system. Statistical anomaly builds two profiles with one consisting of data from a training phase and the other profile is the current state of the computer system. The training state profile is assumed to display normal activity of the computer. The two profiles are compared and if any deviations from the training profile, the current profile is identified as an intrusion [23]. Using a rule based detection method, rules define the thresholds instead of a numerical value. The behavior is defined by rules, which is defined as a set or a sequence of events that describes the behavior of an intrusion [10].

Data-mining can automate the analysis and extract relevant features used for the detection algorithm. The methods developed over the years include classification, clustering, outlier detection and rule based discovery. Data-mining and anomaly detection based methods in general suffers from high false alarm rates. A method used in the past for anomaly detection was using the host's sequences of system calls and modeling the processes and valid system calls they generated [23]. A database of normal-sequences is stored and a sliding time window is used to capture and partition

the data within a frame. The frame that contains the captured sequence of system calls is then compared to the database with the previously established system calls [23].

Limited products are available in the commercial market that implements anomaly detection methods. An IDS using anomaly detection produces a large number of false positives the network security administrator are left with the frustrating task of filter through false positive indications [10]. The real threat are not the vulnerabilities that are found and has the signatures to detect them but the threats that are unseen or the vulnerabilities that are unknown to anyone but the attackers. The signature based systems were really intended for the “script kiddies”, unskilled attackers that rely on known attack tools that usually have signatures [26].

According to Zanero et al [26], anomalous based IDS have been implemented on a purely host based system but have failed when the methodology was used on a network based system with a few exceptions. Zanero claims the reason was because the tendency to produce large volumes of false positives. The outputs to these system does not give any indication on what is wrong but gives a statistical number usually called an anomaly score based on weirdness. One can gather all this data but understanding the set of data that should be monitored is something that researchers have struggled with. Understanding how to describe network flow and model it in an IDS is extremely difficult because of the added constraint to have close to zero false-positive indications [26].

Coal et al. proposed a novel approach detecting masquerading intrusions attacks. Masquerading is a method where an attacker assumes the identity of a legitimate user in a computer system. The detection relies on the signature of a legitimate user which is the sequence of commands it usually initiates. The assumptions made is that a signature captures detectable patterns in a user’s sequence of commands. The a measure of differences between a legitimate user and an attack can be made. Coal’s new algorithm uses pair-wise sequences alignment to characterize similarity between sequences of commands.

Masquerade detection falls under the anomaly based IDS approach but using traditionally anomaly based approach falls short in the detection of masquerading attacks because the anomaly based approach assumes the user is a legitimate user. It cannot differentiate the differences between them because the behavior patterns is usually the same in terms of valid access requirements.

#### *2.4.2 Statistical and Machine Learning Techniques For Intrusion Detection.*

*2.4.2.1 Cluster analysis.* Similarities based on some metric are grouped together and then labeled based on the clusters characteristics. This metric must be chosen carefully as to have correlation of the features and the problem domain. The clusters are usually user defined with knowledge of the solution space giving the algorithm the amount of clusters and the distance criteria each data point must meet to be associated with a cluster.

Clustering has the property to detect outliers of the the data. Outlier detection is an unsupervised learning technique identifying unusual or strange behavior based on every observations. Clustering has the ability to generalize the data. This is useful to counter the polymorphic behavior of hackers and should be able to resist to polymorphic attacks. Clustering can be used on unlabeled data and very powerful in the fact that no priori knowledge is needed for the detection algorithm. Clustering is also adaptive that it can be adjusted or tuned depending on the environment it must work in, that will help reduce the number of false positives.

Zanero et al. [26], proposed a two tier network method to detect anomalies in the flow of the packets on a TCP/IP network. The first tier is where the unsupervised clustering algorithm classifies the payload of the packets, observing and classifying one packet at a time. The system then compresses the information to a single byte. This byte of information is added to the header information of the packet. The second tier then has the packet information and the packet header information to work with.

The second tier process the anomalies seen in the individual packets but also the anomalies seen as a result of the sequence of packets.

The clustering algorithm Zanero used in [26] was  $k$ -means clustering, the principal direction partitioning, and Self Organizing Maps. The new process was tested on the DARPA IDS evaluation 1998 data set. This data set was criticized by many researchers because the data set does not represent realistic network traffic, but was still used as a test the concept of an IDS with a two architecture system. The metric used to measure distances among vectors of data is a simple Euclidean distance function. The proposed architecture has two purposes, intra-packet correlation and inter-packet correlation. The inter-packet is used to detect anomalies in individual packets and the intra-packet is to detect anomalies over a time window that represents distribution of activity and the interactions among sequences of packets. The time window concept of observing past packets was used to model history or memory. Having memory of past packets can increase the ability to detect attacks such as a DOS where checking individual packets would not catch such attack. According Zanero, clustering algorithm could be used for detecting time sequence anomalies by applying them to a rolling window of packets.

Current signature based methods and learning algorithms which rely on labeled data to train, generally cannot detect new intrusions. Portnoy et al [27]. proposed a clustering-based intrusion detection algorithm, which trains the system using unlabeled data. The system would automatically scan the network environment, detect attacks and inform the system administrator to take corrective action. Misuse retraining of a rule base system is done by inserting many labeled instances of new attacks into the data set, and the rule finding algorithm would adjust its rule database accordingly.

According to Portnoy et al. [27], traditional anomaly detection algorithm require the set of the training data to be free of attacks and should be “purely normal data.” The training data if tainted would influence the detection algorithm and not detect



future instance attacks because the system would label similar observed instances as normal. In an ideal world this solution would work as a training set that the system would learn from but getting such a data set is almost impossible to obtain, either labeled data or purely normal data. Most likely of sceneries is that the training set obtained are raw large volumes with intrusion embedded in the system and unlabeled. Portnoy proposes a detection algorithm that is unsupervised taking in as inputs the unlabeled data and from that data attempts to find the intrusions embedded within the data. The intrusions found is then fed to train the detection algorithm. The method used was a clustering algorithm that uses a simple distance based metric. After the clustering of the data the small clusters are labeled as anomalies. The assumption Portnoy has made is that anomalies are rare and therefore would either show up as outliers or within small clusters of the clustering algorithm. Portnoy's detection algorithm first trains the system by finding the clusters of the raw data. The set of clusters of the data is initialized to be empty then for each pass, the data instances are assigned to the closest clusters based on distances for each centroid. The detection phase is to convert an observed instance based on the statistical information of the training set from which the clusters were created. The observed instance is labeled based on the closest cluster set during the training phase.

$k$ -means clustering is one of the more popular methods of clustering data in the field of machine learning and statistics [27]. The algorithm makes passes through the training data and adjusts the cluster's center to the mean of the data points assigned to the each cluster. It will make several passes through the training set until the centers of the clusters stabilizes. This method is computational expensive and is usually used off-line when training the system.

*2.4.2.2 SOM.* According to Wang [28], SOMs is a method used for visualizing data of high dimensionality. This is a clustering algorithm that uses artificial neural networks, the number of clusters is not provided by a user but derived by the algorithm. A SOM consists of nodes and neurons and with each node containing

a weight vector and has the same dimension as the input vector also has the position in the lattice map. The SOM is therefore a mapping from a higher dimensional input space to a lower dimensional map space.

*2.4.2.3 SVM.* According to Shon and Moon the Support Vector Machine (SVM) has proved to be one of the best machine learning algorithms to classify anomalies [4]. Shon and Moon proposed a new SVM approach basically taking the sort-margin SVM and the one class SVM and combining them together to take the advantages of each and eliminates some of the weakness of methods if used alone. They call this new approach Enhanced SVM. The first step to the process is using Self-Organized Feature Map (SOFM) to create a profile of normal packets. They then used a packet filtering scheme based on Passive TCP/IP Fingerprinting (PTF) to reject incomplete network traffic. A Genetic Algorithm (GA) is used for extracting the optimized information from the raw internet packets. Lastly they used the flow of packets based on temporal relationships during data preprocessing. The overall frame work is shown in figure 4.

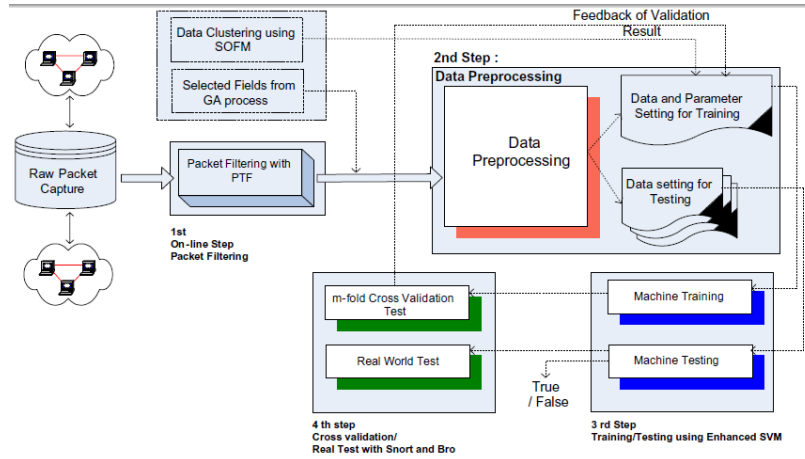


Figure 2.4: Shon and Moon Enhanced SVM approach to an IDS [4]

The SOFM step of Shon and Moon’s proposed process is to normalize and obtain normal data for the training of the system. The normal data is essential for the training of a supervised system but also provides intelligent criterion to the

unsupervised learning of the system. In Shon and Moon’s method, packets for normal learning is needed as a measure of the correct learning have normal characteristics. SOFM aim is to create clusters and distinguish them from with normal features. SOFM is implemented as an unsupervised neural network and converts non-linear statical relationships between data points in high-dimensional space into relationships in a two-dimensional map. SOFM clusters are made in accordance with the attributes of packets.

The TCP/IP fingerprinting purpose is to confirm the structure of the received packets. It discards any packets that is not recognized as a common constructed packet. The fingerprinting uses heuristics and is expected to identify normal systems if fingerprints of the systems are submitted. This does not filter packets with malicious payload but identifies that the packet if on normal construction and not malform.

Moon and Shon used a genetic algorithm (GA) technique for feature selection. The method and the fitness function used is described in [29]. GA is a well known method for finding solutions for optimization problems. GA uses three operators to produce the next generation from the current state of reproduction, crossover, and mutation. Shon and Moon first transform packets into binary gene strings. The initial population consists of a set of randomly generated 24-bit strings, with 13 bits for IP fields and 11 bits for TCP fields.

Two methods of SVM can be used for machine learning, the supervised approach also called Soft Margin SVM or the unsupervised approach also called One-class SVM [4]. The supervised approach has a purpose to decrease misclassified data. A slack variable is used to decrease misclassification. A single class learning for classifying outliers can be used with the unsupervised approach.

The unsupervised version of the SVM algorithm identifies outliers among the normal class then use these outliers to define the anomaly class. The algorithm maps the data in a feature space  $H$  using an appropriate kernel function, and then

attempts to find the hyper-plane that separates the mapped vectors from the origin with maximum margin.

Shon and Moon make the assumption similar to past research that anomalous activity is small compared to the normal traffic in any training or real world data. The Enhanced SVM approach proposed by Shon and Moon is a method that has the capability of one-class SVM through modifying the soft margin SVM. The complete details of the enhanced SVM could be found in [29]. The data processing step is to take into account temporal event sequences. According to Shon and Moon picking out relationships among the sequences of network packets the knowledge would deeply influence the performance of the learning. To capture this a sliding window concept was used.

*2.4.2.4 Bayesian Networks.* Bayesian networks are graphs that represents transition rules and their probabilistic interdependencies [3]. The nodes represents the state of random variables with the conditional probability table that determine state given the state of the parents.

*2.4.2.5 State Transition Table.* A state transition table is a table of sequence of behaviors or actions an intruder does [3]. The detection of an intruder is a matching of these sequences of actions to observed sequences of actions.

*2.4.2.6 Artificial Neural Networks (ANN).* Artificial Neural Networks (ANN) is a system that is trained to recognize patterns from input data and associate the input pattern to an outcome [3].

*2.4.2.7 Genetic Algorithm (GA).* Genetic Algorithm (GA) is search algorithm that can search for solutions in a large search space. The mechanism of the search is modeled after natural reproduction system in nature. The fittest of characteristics survives upon generations after random changes of reproduction is placed. In IDSs GSs involved evolving signatures on intrusions [3].

Intrusion detection methods should be robust, adaptive and efficient. Hansen et al, suggests that genetic programming would give you this. Genetic programming is a subset of the famous and well known GA. Hansen and his group worked on a new method using machine-coded linear genomes and a homologous crossover operator.

*2.4.2.8 Artificial Immune System .* Artificial immune systems are systems that mimic the biology of the human immune system to attack foreign objects within itself. The system have agents like white blood cells of the human body and will detect and attack foreign objects. In a computer system it neutralized potential intruders by identifying if that foreign object is self or non-self item and then take actions to neutralize or eliminate the non-self object.

*2.4.3 Network based intrusion detection systems (NIDS).* The most popular IDS systems are in the form of NIDS. The majority of the commercial market uses NIDS technology and methods. [10] IDSs are classified into network based or host based and differ in the data they analyze. A network based IDS analyzes network traffic. It looks at network packet data to detect the evidence of intrusion. Host based IDS analyzes internal information and find intrusion evidence in operating system such as logs. [9]

There are many advantages to NIDS, one is the ability to monitor more than one machine where HIDS has to be individual installed on every machine it protects on the network. NIDS protects the network it is monitoring and does not need to be installed on any running existing system. The system can be external with minimal down time to the network with respect deployment and installation [10] Some of the disadvantages of NIDS are there is no guarantee that all the packets of the network traffic is captured by the NIDS and can possibly miss attacks. NIDS uses network traffic such as network packets either header information, payload data or both. Another disadvantage is if the network is using encryption, the IDS will be ineffective because NIDS cannot decrypted the data at the network level. [10]

The speed and the amount of data in today's network environment makes capturing 100% of the incoming packets unfeasible. In [30], Sperotto et al. presents a different approach of NIDS. Monitoring the "Flow" of a network and not the individual packet payload. There are measurement systems that can monitor and give information of the flow of the network some of them are Netflow and IPFIX [30]. These tools can be used as a compliment system to the NIDS. Sperotto's model is a combination of both ideas of network monitoring and data payload monitoring. The model has two processes with the first monitoring packet payload and the second monitoring of the network flow. The model is defined as a two step method with the first step extracting information from packets and network traffic to create a data structure called a "flow." In IPFIX a flow is in the form of, (*source address, destination address, source port, destination port, and protocol*). The second step is collecting the flows created to be used by another process to analyze the "flows."

A subset of NIDS that detects collaborative attacks and intrusions is described in [31]. These systems called collaborative intrusion detection systems (CIDS) rely on multiple data sources instead of single node on the network or a single target host computer. Systems like CIDS have the potential of detecting coordinated attacks such as, stealthy scans, worms, and distributed denial-of-service (DDoS) attack that would make regular isolated NIDS unable to detect these types of attack because of the single source of data or a portion of the internet being observed. Stealthy scans, worms and DDos are the predominant collaborative attacks used today [31]. The stealthy scan is a reconnaissance type of attack that will look for targets to attack and use the vulnerabilities found in the stealthy scan to control the target computer.

Zanero et. al [26] proposes a two tier architecture to overcome the problem of computational workload, using data mining techniques on network traffic data. The first tier is an unsupervised clustering algorithm that reduces the network packets payload to a tractable size. The second tier is the anomaly detection algorithm. The first tier function is to reduced the packets payload to a tractable size so the detection algorithms has the packets payload data to be used in its detection algorithm. The

two tier architecture has made it possible for the detection algorithm to use payload data where traditional network intrusion detection system would discard payload data and would analyze only packet header information.

Collecting intrusion data is difficult because of the scarce nature of the data. What Yeung and Chow proposed to overcome the lack of intrusive data for training is to take a nonparametric density estimation approach based on the Parzen-window estimators with Gaussian kernels to build an intrusion detection using only normal data [32]. Essentially the intrusion detection problem could be simplified to a data mining approach by “mining” through data either host or network to detect possible attack from intruders

What Yueng and Chow used in their method was to use density estimation which assumes a probabilistic generative model for the observed data. Density estimation is the process of estimating the underlying density function for the gathered training data. This model is then used to detect novel attacks. The density function using parametric density estimation was too restrictive so they decided to use a non-parametric method for the estimation more specifically the Parzen-Window Density Estimation.

To test their novelty detection using the Parzen-window estimation they used hypothesis testing. If  $\omega_0$  represents anomaly and  $\omega_1$  represents normal behavior of the system then their prior probabilities are denoted as  $P(\omega_0)$  and  $P(\omega_1)$  with the probability density functions noted as  $p(x|\omega_0)$  and  $p(x|\omega_1)$ . Using Bayes theorem  $x \in \omega_1$  if  $p(x|\omega_1) > p(x|\omega_0)(P(\omega_0)/P(\omega_1))$ . The difficulty of this approach is that it is very difficult to model the distribution of anomalous events. A solution they came up with is to use hypothesis testing. Given an unknown case  $x$  from the dataset, the goal is to decide if  $x \in \omega_1$  the anomalous set. The disadvantage is the construction of the model has a high computational demand. According to the Young and Chow the method has characteristics similar to K-nearest-neighbor classifier with similar results.

Li and Guo [33] proposes a novel supervised network intrusion detection method based on TCM-KNN (Transductive Confidence Machine for K-Nearest Neighbor) machine learning algorithm. There are two major ideas to the method Li and Guo has proposed one is that they used what they call active learning where the system would select much fewer good quality data for the training instead of just blindly and randomly picking data for training. The second is the use of feature selection to reduce computation and speeding up the classification step. Using the  $p - value$ , the probability of observing a point in the sample space that can be considered more extreme than a sample of data. It is a measure of how well the data supports the null hypothesis basically the point belongs to a particular class. The smaller the value the greater the chance the point of data is an outlier. The transductive test in Li and Guo's method is also known the strangeness measure and corresponds to the uncertainty of the data point. The details of the approach could be found in [33]

The principle idea of this method is training the system with the labeled data provided them a systematically selecting unlabeled data to classify. The new unlabeled data is then used to help with the training of the system and is part of the labeled training data in the future.

*2.4.4 Host based intrusion detection systems (HIDS).* HIDS is a class of IDS that uses information gathered by the host operating system. The operating system is considered a trusted entity and the data extracted is used to detect the evidence of intrusion. Vigna describes two layers where HIDS could be implemented, the operating system-level and the application level. [9] An advantage of host based intrusion detection system is the visibility of application interactions between users and applications or between applications. With encrypted communication an NIDS monitoring network traffic using signature or anomaly detection would be ineffective because the data captured would be unreadable. HIDS running on the host will see the data un-encrypted, the data in the application level will decrypted the data so the host application can use the information.



HIDS has the ability to monitor the host in the application level where the data stream is un-encrypted. Some of the disadvantages of HIDS is the scalability of the system. HIDS requires to be installed on every target computer in the organizations network for the whole network to be protected. If the hardware is different then the HIDS is individually configured with respect to the individual hardware, software and type of user [10]. An HIDS can be a targeted for an attack. Usually HIDS is a piece of software running on the host and can be tampered with if an intruder penetrated the system before any type of detection [10]. An HIDS can starve other processes running on the computer if the HIDS has a heavy computational load therefor influencing the performance of the whole system [10]. The activities that are monitored using a HIDS are CPU information, network connections and process [23].

A promising area in HIDS is monitoring the system calls of the host. Tandon et al. proposes a method for intrusion detection using system calls [14]. The method uses sequences of system calls as well as their arguments. The addition of system call arguments for IDS analysis in combination with sequences of system calls minimizes attack using methods that evade IDS technology. Mimicry attacks are intrusion methods where inserting dummy system calls with invalid arguments form a legitimate sequence of system calls making the attack undetected [14]. The method Tandon et al. describes is using a fixed length sliding window of the sequence for the system calls [14]. A threshold is limit or the amount of anomalous activity the system cannot exceed. An anomalous activity is defined as deviations from a sequence of events describing normal sequences within the given time window. This time window slides across the length of recorded time looking for anomalous activity [14].

A host running a HIDS is vulnerable to attacks where the target of the attack is the HIDS application. An attacker can penetrate the system, disable the HIDS before being detected. This leads to methods being developed for protecting the HIDS from attack. Laureano et al. introduces a model that implements HIDS using a virtual machine that shields the HIDS from attack [5]. Virtual machines can sand box its operations making the host operating system unaffected with the malicious activity

inside the virtual machine. The proposed architecture utilizes the separation virtual machine provides the execution of processes. The advantage of this architecture is the sensor monitoring for attacks is invisible to the attacker. The idea of separability of process execution in one machine adds a secondary but equally important aspect of a virtual machine and that is monitoring malicious activity without being affected. The observations could be used as a learning mechanism learning the behavioral activity of real malicious activity in the wild [5].

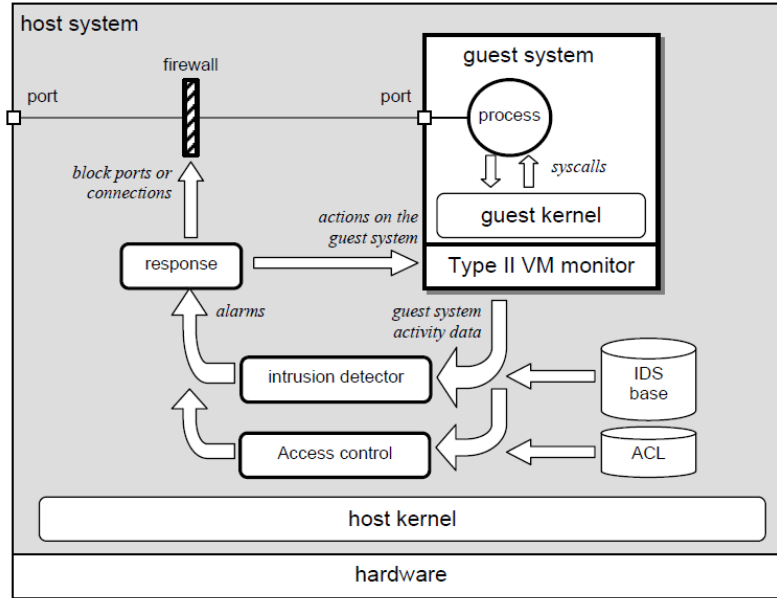


Figure 2.5: Laureano, et al. Model for protecting HIDS sensors using virtual machines [5].

In figure 2.6, the outside traffic does not reach the host operating system and all incoming traffic is funneled to the guest operating system or virtual machine. The functions of the HIDS is located outside the guest system and out of reach from intruders with intent to tamper the IDS. The data stream flowing out of the virtual machine has the activity data from the guest operating system. The collected data is compared to previously learned model from the past where model characterizes past attacks and therefore could find intrusions in the guest operating system [5]. Laureano implements this IDS using virtual machine with a knowledge base using misuse detection. The primary goal was not to create a new method of HIDS but

encapsulate the HIDS itself and protect it from attack. The author also proposed a method of detecting invalid sequences of system call with a certain size of length  $k$  and integrity checking the ACL (access control list) from a past ACL. The model has two operating modes: a learning and a monitoring mode [5]. The learning mode learns the normal behavior of the system in terms of sequences of system calls and ACL. The monitoring phase monitors and compares current virtual machine behavior profiles from the knowledge base of system call sequence and ACL.

Malan et al. [6] proposed HIDS implementation for worms on a network. The solution the author explains in [6] is a collaborative solution where every machine on the network works together to find similarities that would indicate a worm. As shown in figure 2.7 each machine is connected to a processing computer node that collects behavioral snapshots of computers on the network. Depending on the size of the network there might be multiple processing computers interconnected together. The author argues that because it is very unlikely similar behavior are observed across the whole network unless the network has been infected by a worm. The fast spread of a worm is a characteristic of similar behavior across the network and implies that a worm is spreading on the network. The snapshots from machines is sent to a snapshot server. The snapshots are behavior patterns of system calls executed in a given time window [6]. Malan's definition of a behavior of a system is a sequence of system calls and from that he defines an anomaly as a high correlation between behavior across the network.

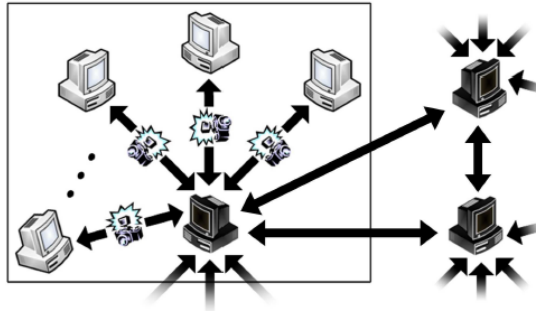


Figure 2.6: Malan et, al. Model HIDS for detection of worms [6].

Large amount of false positives alerts has lead researchers developing methods to reduce them. Finding that perfect model of normal behavior has lead to data reduction techniques were redundant data can hurt the models accuracy. Another approach is to eliminate redundant alerts. A method Feng et al. [7] introduces is a fusion of alerts to reduce the redundant alerts the system outputs. Feng implements this model using a Subject-Verb-Object (SVO) structure with alert fusion and alert verification modules. The system has many alerts outputs from different components of the SVO system and the alert fusion module will fuse alerts that are identical and verify output alerts are legitimate positive alert. The proposed model is shown in figure 2.8.

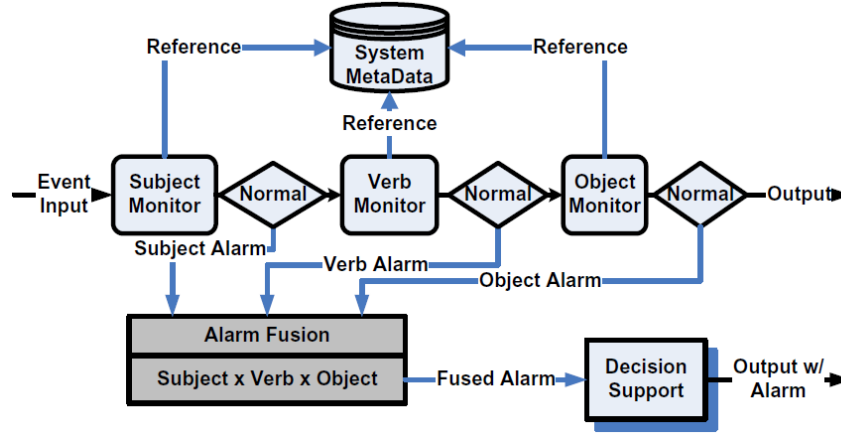


Figure 2.7: Feng, et al. HIDS model with alert fusion [7].

The inputs to this model is an event and then a three layer individual process analyzes that event. The Subject Monitor analyzes the subject of the event also referred to the originator of the action, the Verb Monitor analyzes the verb of the event defined as the action the subject initiated and the Object Monitor analyzes the object denoted as some entity involved in the subject's action. The first layer is the Subject layer if the Subject Monitor finds no subject defined anomalous activity then the event is forwarded to the Verb Monitor if no verb defined anomalous activity the event is forwarded to the Object Monitor and if no object defined anomalous activity then the event is not anomalous. At any point in the three monitor modules finds

anomalous activity then an alarm is sent to the Alarm Fusion module. The purpose of the Alarm Fusion module is to eliminate alarms. If the Subject sends an alarm to the Alarm Fusion module and the Verb Monitor sends the same alarm the Alarm Fusion module, it combines the two alarms if it finds those alarm identical.

In [23], Hoang et al. implemented a hybrid anomaly IDS using features from system calls of programs. Hoag combined two methods, one using a hidden Markov model (HMM) and the other a using a database of valid system call sequences. The output of the two detection schemes was then fed into a fuzzy-based logic rule based system used to infer a boundary between anomalous and and normal behavior. The training process proposed by Hoang, is to divide the a single observation  $O$  into  $K$  sub-sequences.

The fuzzy-based detection schemes consists of two stages, the training stage and the testing stage. The training stage is where the the database of the sequence of system calls and the HMM model is built. What the database contains is the list of all short sequence of system calls found in the training data. Each sequence is of size  $k$  system calls long with the frequency also provided by this database. The HMM is trained in the training method discussed above based on the incremental training scheme [23], with the fuzzy rules defined in the training step as well. In the testing phase a sliding window method is used to extract the sequences of system calls called a short sequence. The short sequence is fed into the HMM model and the database of allowable short sequence system calls. The output is used to compute the parameters of the fuzzy inference system. The fuzzy system classifies the observed data based on the outputs of the HMM [23]. Hoang, manually devised 17 fuzzy rules for the sequence classification. An example on of the rule is: “IF probability IS Low AND distance IS Zero AND frequency IS Low THEN the test sequence IS abnormal.” where the data being fed into the fuzzy rule sets are {probability, distance, frequency, and short sequence of system calls}.

A subset of host based IDS is application based IDS where the collection of application data is collected from individual applications running on the host. The usual mechanism of data collection of application information is within the application logs the operating system provides. [10] This method has great potential in detecting insider threat attacks, where an authorized user with suspicious activity will set an alarm by the IDS if the authorized user somehow raised it's access in the system. This type of information is available because of the monitoring in the application layer where dependencies are mapped from application, process and users. Some of the advantages of application based IDS are the clear mapping of interactions between users and applications. The ability to correlate and match the unauthorized activity to the user makes this intrusion detection method very powerful, it goes beyond just setting an alarm when an intrusion has occurred. If the network traffic is encrypted an application based IDS would still be effective because the data is gathered at the application layer. The data is usually decrypted at this point so the application could use the data. [10] The drawbacks is it has the same vulnerabilities of HIDS [10]. Unlike the audit trails the operating system provides the information the application generates is not as protected. The audit-trails and logs are usually trusted entities because the information comes from kernel level but the information in application based IDS is generated from user level and can easily modified.

## ***2.5 Flow-Based Features***

Moore et. al [34] developed a feature set for use in flow-based network classification where a flow is defined as one or more packets traveling between two computer. He provided the features to characterize network flows using statistical information about packet length, inter-packet timing, and TCP information. The packet statistics used deal with packets and packet headers calculating simple statistics of the features of the minimum, quartiles, mean, variance, maximum and the fast Fourier transform of the data. This thesis was to take this approach of net workflows and apply these

simple statistics and characterize processes in memory. The statistical feature set is described in chapter 3.

*2.5.1 Network Based Flow-Based Features.* A method described in Duffield’s et al. [35] paper is a packet classification method that uses rule based for identifying traffic anomalies. Duffield’s paper explored the use of correlations between packet and flow level information. The payload information is ignored and the classification of anomalous activity is entirely based on network flow. The goal of Duffield’s approach is to detect network attack traffic. A tool like Snort can run on any general purpose computer with rules created by the user or from a community source that can be downloaded. In an ideal world with infinite storage and infinite computation with no possibility of over working the system an IDS should look at every single packet payload and header but because there are resource and computational limitations this method would be impractical. Duffield’s plan was to use signature based system on network flow data unlike what Snort that looks at signatures individual packets either header or payload information. Duffield’s states that “Flow statistics are compact and collected ubiquitously with most ISP’s networks.”

Duffield constructed rules at the flow level that reproduce the action rules at the packet-level rules. A machine learning algorithm was used to detect unwanted traffic using flow signatures. Duffield’s proposed method is different from the traditional unsupervised learning methods such as Bayesian networks, PCA and clustering in the fact that rather than alarming unknown unusual events based on deviation from the normal behavior model. Duffield uses the set of events alerted by packet rules as representing the most complete available knowledge. This learned knowledge is then used as the input into the machine learning algorithm to reproduce the alerts at the flow level.

Duffield separates predicates representing the different data sources and combine them using logical AND and OR operations. The different predicates are the flow-header, packet-payload, and meta-information.

## **2.6 Summary**

This chapter has presented the background information for machine learning, and IDS. The problem for detecting malicious activity using signature or anomaly based techniques. Chapter 3 builds on work from past research for detecting anomalous behavior using flow based features of network data and applying it to flows of processes. A new set of features for host based anomaly detection is presented. Finally, Chapter 3 also provides the method for clustering process flows for use in the identification of anomalous processes.



### III. Methodology

This chapter describes the research methodology to evaluate the novel use of host based process flow feature clustering to detect malicious processes using VMI. The system is categorized as Host-Based detection with the data of interest as memory features. CMAT provides the VMI sensor, capable of live memory captures of a system. It works by capturing the host OS memory while operating in a hyper visor environment, producing 5 feature files approximately every 30 seconds and 1 file that relates to registry information every 30 minutes.

The output files of CMAT are preprocessed using a sliding window to extract features across the life of all processes between a start time ( $t_{start}$ ) and an end time ( $t_{end}$ ). These windows of observations of processes are the features used to cluster and identify groups that represents process type. These groups are used to identify new flows for possible membership to one or more of the groups in the decision space.

This chapter details the data collection process, which is followed by a description of how the collected data was injected into a database. Using the models developed from the clustering algorithm we systematically test the models using labeled data captured from normal scenarios with several different types of applications running. The goal is to determine if new instances of process flows are correctly identified by the clusters. These groups are the knowledge developed from the computer domain that give labels to the types of applications associated as members.

#### 3.1 Problem Definition

Memory is volatile and provides the most accurate snapshot of the computer state at a moment in time [36]. A program runs in memory, and a sensor like CMAT makes it possible to observe the connection the process has to other system resources. The resource of interest include process ID number, application name, username, local ports, remote ports, dynamic link libraries, permissions, file handles, registry and system drivers information. The feature files CMAT generates may be used to correlate anomalous processes. The problem with using the raw feature files to find

connections to anomalous activity is that the amount of data is too large to manually analyze for any correlation to malicious processes. The evidence of intrusions or malicious activity are assumed to be hidden within the CMAT raw feature files. A computer administrator just has to find it. Correlating data from the raw feature files to malicious processes is difficult to do in a timely manner. A transformation to the raw files is made for a better understanding of the types of processes and its behavior.

An abstraction to features of processes called flows are made. These flows are grouped together providing a label that describes the application group by type or behavior. A relationship is made to individual processes to its behavior based on membership. Grouping similar processes together provides the means for finding potential threats. New unseen flows are labeled based on nearest group using the Euclidian distance measure. An anomalous flow is an instance outside a threshold of the given group. For example, a “calc.exe” should be nearest to the group of graphical user interface(GUI) only processes that has ‘calc.exe’ as part of its membership list. If a process flow identifies itself as “calc.exe” and is outside this group’s threshold, that particular flow is identified as anomalous. The hypothesis for this research is that the discovery of groupings of processes executing in memory can be identified.

*3.1.1 Research Goals.* In the field a computer forensics, incident response is the preliminary stage of an investigation where the goal is to gather as much data to determine whether or not an unlawful, unauthorized activity has occurred. Forensic response is used by law enforcement to investigate crimes where a computer is part of the crime scene, or is searched for possible evidence. The primary goal for this research is to leverage the host features the computer forensic community uses in their initial response investigation and monitor those features gathering statistical information and possibly detect anomalous activity.

In [34], a set of features was described to the public for use in research in the field of classifying real network data. Moore [34], defines a network flow as flows of network packets from a source to a destination. Statical features are calculated with

respect to packet communication. The features could be found in [34], and contain simple statistics such as mean, quartile, median, minimum, maximum, variance and also the discrete fast fourier transform of packet frequencies. This research combines host features used in computers forensics to calculate simple statistics similar to the network flow implementation. Instead of statistics of network packets this research uses statistics of processes. The features are calculated from a sensor that monitors processes information in memory. A host based process flow is defined the change in processes flow features over the change in time.

The processes flows are clustered together to be used for association of new flows to these predefined clusters. This research identifies the amount of clusters needed to accurately model application types the window of observations that best describes a processes flow. The goal is to determine if a Host-Based Anomaly detection method may be developed using statistical running processes information from memory analysis. The Host-Based Anomaly detection method classifies the dynamic behavior of a process called a “flow” into their application types. An anomalous process is characterized as a processes outside the threshold of its group. A learning step is used to train and these grouping and determine a threshold value. Labeled data was used to generate the models and verify if anomalous events are identified.

*3.1.2 Expected Outcome.* After this proof-of-concept method to detect anomalous processes is developed and tested, an organization could possibly integrate this system in a virtual machine environment. The current CMAT software can capture live memory only via a virtual machine interface but CMAT software could be updated to run and capture memory from the host machine. Detection of anomalous processes could be used to safeguard against attackers and malicious software.

*3.1.3 Assumptions.* The fundamental assumptions made in this research are computers are running on a virtual environment and the CMAT sensor is running on the host OS capturing live memory. An anomalous process is a processes with statistical characteristics outside what is defined as normal and in this case its the

distribution of “flows” and its respected classification. The classification is the application type of a process. Past research [20] [21], used incident response toolkit tools for the host sensors.

### ***3.2 Research Approach***

Li et al., [37] researched using network-traffic features for classification of network-based application using statistical behavioral of network flow and Deep-Packet Inspection. Li et al., [37] classified application types based on network features. The dynamic process behavior called “flows” are the data points the classifier uses to classify the associated application that is correlated to the process. This research examines applying the idea of flow features to host process data calculate dynamic statistics of processes in memory and determine if a classification model could be developed. By accurately classifying application types, process flows could further be used to detect anomalous behavior and indicate the presence of malicious software or an attacker in the system.

### ***3.3 Host Based Process Flow Features***

CMAT provides one tool to capture all the feature needed for this proposed method for detection of anomalous processes flows. The features of interest are the features used in the field of digital forensic for situational awareness. Past research used several tools, combining all output from the tools and feeding it into their proposed detection method. The incident response tools utilizes trusted files that execute from a CD-ROM or DVD. GUI driven tools may interact with the machine and affect evidence collection and therefore should utilize command-line tools. Incident response tools may is used as a snap shot sensor to give the state of the machine at a point in time. The outputs of these snap shot sensors are fed into an IDS to detect anomalous activity. A transformation from unusable data with too much detail for any situational awareness understanding to instances of data elements used for correlating anomalous events. CMAT raw files are collected and features are calculated for each

process for a given time window. The following section describes the features being extracted in a computer system, the snapshot sensor that is capable of capturing the features and CMAT's capability of capturing the equivalent features of the snapshot sensor.

Instead of using tools that run on the same level the system it was intended protect, this research took the approach of taking the sensor and placing it outside of the operating system. The environment is therefore a virtual machine with the sensor running on the host operating system capturing features on the guest operating system. CMAT provides this capability and the features it provides are process IDs, application names, DLLs, mapped address spaces and registry key information. The assumption here is that CMAT can provide most of snap shot sensor capabilities so CMAT can be used as the sole host sensor to an IDS. The benefit from CMAT to the other sensor tools described above is the security it can provide because it executes outside of the operating system making it harder get compromised or detected by intruders.

*3.3.1 Compiled Memory Analysis Tool (CMAT).* The Compiled Memory Analysis Tool(CMAT) is a forensic analysis tool that extracts features from memory for situational awareness. The softwares can work in two environments:

1. Host O/S: Windows 7 64 bit O/S Memory Dumps: Windows 7 64 bit, Windows 7 32 bit, Windows Vista 64 bit, Windows Vista 32 bit, Windows Server 03 64 bit, Windows Server 03 32 bit, Windows XP 64 bit, Windows XP 32 bit
2. Host O/S: CentOS 32 bit O/S Hypervisor: Xen Guest O/S: Windows 7 32 bit, Windows XP 32 bit Guest O/S accessed via XenAccess

The data collected from memory by CMAT are processes and their associated features. CMAT provides one tool as a sensor to processes of the system where past research [21] [20] relied on several smaller software tools to gather the same features. The following sections describe the software tools past research used in combinations

to extract features of processes. The tools would be running simultaneously extracting the features. The assumption here is CMAT features is easier to correlate processes with their features. The other tools are separate programs that have no interactions with each other. Past research had to work with several sensors and linking them in a way to gather useful information.

*3.3.1.1 Processes Overview.* To capture processes information, PsList can be used. This tool is part of the family of Windows SysInternal tools available as a troubleshooting utility at the Microsoft TechNet website. The features this tool provides are process ID, process name, user/kernel time, virtual memory sizes, working set, private virtual memory, private virtual memory peak and page faults. This tool can be configured to output the features to a text file at whatever frequency desired.

*3.3.1.2 Dynamic Link Libraries (DLL).* Processes are usually associated with DLLs. ListDLLs.exe is another tool from the family of Windows SysInternal tools. This tool provides the full path names of the loaded modules associated to their processes.

*3.3.1.3 Users and Session.* A sensor that monitors logged on user times and processes that are associated with a particular user is logonsession.exe. This tool is also part of the family of Windows SysInternal tools.

*3.3.1.4 Network Connections.* A sensor that monitors network connections is NetStat.exe also part of the family of Windows SysInternal tools. This tool captures active TCP connections, ports their associated process IDs, network statistics (IPv4 statistics and IPv6 statistics), and IP routing table information.

*3.3.2 Data Preparation.* The following subsections outline the procedures used to extract, parse and store the CMAT data for feature generation and save the data into a comma separated file (CSV). The data contained in these CSV files

Table 3.1: The Database Tables Created from the CMAT Feature Files Output

Database Table	Table COLUMNS
Table 1	MachineID,CollectTime, PID, IPv6, LocalPort, RemotePort, Protocol
Table 2	MachineID, CollectTime, PID, BaseDLL, FullDLL
Table 3	MachineID, CollectTime, PID, Application Name, User
Table 4	MachineID, CollectTime, PID, Permissions
Table 5	MachineID, CollectTime, PID, Registry Key Value
Table 6	MachineID, CollectTime, Driver Name, Image Base

Table 3.2: Raw CMAT Output Files

FILE 1	FILE 2	FILE 3
Process ID	Process ID	Process ID
Process name	Local IP Address/Port	Base DLL Name
Username	Remote IP Address/Port	Full DLL Name
	IPv6{Boolean}	
FILE 4	FILE 5	FILE 6
Process ID	Process ID	Driver Name
Memory address of file	Memory address of registry	Memory address of driver
Permissions of Process ID	HIVE/Cell name	

were injected into a MySQL database and the database was used as an interface to generate the process flow features at an instance of time  $t$ . The tables and their columns generated from the CMAT output files are shown in Table 3.1. The 6 tables were generated after the features from the CMAT files were parsed and injected into these tables. The 6 tables were used to generate a process flow table that captures the behavior of processes from a given instance of time. Querying the table that represents the processes behavior at a given instance in time was then used to create processes flow for each processes. This new table is the dynamic behavior of a processes from an observed window of time.

The naming convention for the six files generated for each CMAT output twice per minute for each file name the ‘xx’ is the first two characters the next 16 characters is the time in a format of :{year,month,hour,minute,second}. It is followed by the file number described above and the file type, a ‘.txt’ file. Each CMAT file that was generated by a particular machine was grouped together into folders labeled with the

machine name that produced the files. Each host has a numbered folder with the team name, for each folder are subfolders for each VM and their CMAT output files.

The features in the six text files are extracted and imported into a MySQL database for preprocessing. A sliding window with overlapping observations of the was imposed upon the data to map each instance of the observed window into feature space where a clustering algorithm can learn and build a model of normal process behavior.

To create the process flows then learning from the flows by using a clustering algorithm is one of the goals of this research. Host based anomaly detection in the past focused on audit logs as the basis of the data analysis to detect anomalous behavior [13]. While network based anomaly detection focused on packet payload. An alternative was a method that does not require packet payload information, but to keep track of network flows of packets within a network. In theory this method would have less of a computational burden because it does not require all packet payload information to be captured and analyzed. What this research tries to accomplish is to have a host based anomaly detection method that does not rely on audit logs for feature extraction to track the host system state but instead to track each processes behavior in memory. An alarm is set if an anomalous process is behaving abnormal.

*3.3.3 Host-Based Process Flow Feature Generation.* In a network, a flow is described as network packet traffic information that describes the flow of traffic from a source to a destination. The changes to these flows is one set of feature anomaly based IDSs use to classify network as malicious or not.

Instead of using the flow of packets in a network this research focuses on the flow of the processes internal features described above. The flows of each process in memory are the features that are extracted for this research. The flows of a process are the features with characterization statistics in a observed time window. Statistical information is generated from the time window with respect to the processes because



Table 3.3: Process Flow Characteristics

PID	#APPLICATIONS
#TCP	#READ PERMISSIONS
#UDP	#WRITE PERMISSIONS
#IPv6	#DELETE PERMISSIONS
#DLLS	#NO PERMISSIONS
#USERS	#REGISTRY KEYS
	#DRIVERS

a time window gives multiple discrete observed instances of the same processes. One could extract statistical information of the observed process in the given time window.

*3.3.4 Process Flow and Features.* A process flow is the dynamic characteristics of a process from a single computer in an observation window from  $t_0$  to  $t_1$ . The features extracted from the 6 files in an instance of time  $t$  are described as the counts of each feature associated with their PIDs.

Applying an observation window would give the dynamic behavior of a process from a starting time  $t_0$  to an ending time  $t_1$ . This observations are called process flows. Extracting all the flows from the CMAT output files, then clustering these process flows would result in a classifier that would give the generalized model of a processes and its application characteristics. New unlabeled captured data is captured and preprocessed to extract the process flows so a processes can be classified to the nearest distance from a cluster. A process flow with an observation window  $t_0$  to  $t_1$  is described as the features and their characteristic statics of each feature.

A new table was generated where each row is a process flow that describes a PID in a given time window. The 194 features are the columns in Table 3.4. Each discrete process flow features at time  $t$  were put into a new table corresponding to one row in that table. That new table was used to generate a process flow from a beginning time  $t_0$  to an end time  $t_1$ . For example if our discrete times in the process flow features table are 1000, 1030, 1100, 1130, 1200, 1230, 1300, 1330 and our PIDs associated for the discrete times are:  $\{1,2,3\}, \{3,5,4\}, \{1,9\}, \{1,4,9,113\}, \{5,6\}, \{2,4,6,7\}, \{0,7\}, \{9\}$  then in the first row in in the process flow features table are features corresponding with PID

Table 3.4: An instance of Processes Flow and Features

Columns	Feature Description
1	Windownumber = $(1, 2, 3, \dots, n \ (t_{beginningWindow} - t_{end\_window}))$
2	PID = The process IDs for each process observed
3 - 14	At $T_0$ the observation counts for Ports, TCP, UDP, DLLs, Users, Applications, Read Permissions, Write Permissions, Delete Permissions, No Permissions, Registry Keys and System Drivers.
15 - 26	At $T_1$ the observation counts for Ports, TCP, UDP, DLLs, Users, Applications, Read Permissions, Write Permissions, Delete Permissions, No Permissions, Registry Keys and System Drivers.
27 - 38	The total observation (a unique window number) of all counts from $T_0$ to $T_1$ of the following features: Ports, TCP, UDP, DLLs, Users, Applications, Read Permissions, Write Permissions, Delete Permissions, No Permissions, Registry Keys and System Drivers.
39 - 50	The maximum observed counts for the following features: Ports, TCP, UDP, DLLs, Users, Applications, Read Permissions, Write Permissions, Delete Permissions, No Permissions, Registry Keys and System Drivers.
51 - 62	The minimum observed counts for the following features: Ports, TCP, UDP, DLLs, Users, Applications, Read Permissions, Write Permissions, Delete Permissions, No Permissions, Registry Keys and System Drivers.
63 - 74	The average counts for Ports, TCP, UDP, DLLs, Users, Applications, Read Permissions, Write Permissions, Delete Permissions, No Permissions, Registry Keys and System Drivers in the time window
75 - 86	The number of counts for Ports, TCP, UDP, DLLs, Users, Applications, Read Permissions, Write Permissions, Delete Permissions, No Permissions, Registry Keys and System Drivers of the first quartile of the time window
87 - 98	The median count number for a given time window for Ports, TCP, UDP, DLLs, Users, Applications, Read Permissions, Write Permissions, Delete Permissions, No Permissions, Registry Keys and System Drivers.
99 - 110	The number of counts for Ports, TCP, UDP, DLLs, Users, Applications, Read Permissions, Write Permissions, Delete Permissions, No Permissions, Registry Keys and System Drivers of the third quartile of the time window.
111 - 122	The variance in a time window of counts for Ports, TCP, UDP, DLLs, Users, Applications, Read Permissions, Write Permissions, Delete Permissions, No Permissions, Registry Keys and System Drivers.
123 - 134	The standard deviation in a time window of counts for Ports, TCP, UDP, DLLs, Users, Applications, Read Permissions, Write Permissions, Delete Permissions, No Permissions, Registry Keys and System Drivers.
135 - 194	The 5 highest values for each of the inputs of counts of the time window for Ports, TCP, UDP, DLLs, Users, Applications, Read Permissions, Write Permissions, Delete Permissions, No Permissions, Registry Keys and System Drivers after the array of values is transform to another domain by the FFT algorithm.

= 1 with time 1000 and every columns in that row are the counts of the PID characteristics {TCP, UDP, IPv6, DLLS, USERS, APPLICATIONS, READ PERMISSIONS, WRITE PERMISSIONS, DELETE PERMISSIONS, NO PERMISSIONS, REGISTRY KEYS, DRIVERS}. The table was generated by inserting a row for every PID and their discrete time windows with all the column features. A flow is then generated by extracting the statistical features of the processes and its given time window. For example if the observed window is 4 then after all the features {TCP, UDP, IPv6, DLLS, USERS, APPLICATIONS, READ PERMISSIONS, WRITE PERMISSIONS, DELETE PERMISSIONS, NO PERMISSIONS, REGISTRY KEYS, DRIVERS} for PID = 1 for all the discrete times were collected then the 190 flow features is generated for PID = 1 for times 1000 to 1130 and that represents one flow for PID = 1.

### 3.4 Clustering

The flows generated from the CMAT output text files were the samples used in the clustering algorithm implemented in the Weka suite [38] of machine learning software. Ideally the clusters group processes based on the similarities of behavior with a definite distinctions between them. The behavior of these objects are the features or attributes that statistically describing the object within an observed time window. The clusters represents the application-centric grouping where an outlier defined as an object associated within a clusters but is outside the threshold of the standard deviation of the center of the cluster's centroid. Finding the outliers can be used as input into an anomaly based IDS where outliers are anomalous processes that can be assumed as an intrusion. This research is primarily focused in classifying processes into application centric groups using purely statistical features observed in a time window bounded by a starting time  $t_{beg}$  to an end time  $t_{end}$ . The model generated could be used in future software development implementing a host based anomaly IDS, setting an alarm when an anomalous process is detected.

Witten, et al. [39] describe clustering as a machine learning technique that divide instances in natural groups. In this research the instances or objects used as input into the clustering algorithm are the processes flows. Each flow would be divided into natural groups. The natural groups are the applications these processes belong based on the statistical behavior characterized within their features. Clustering outputs the groups and the processes assigned to a particular group. The group can be defined as the class where a similar processes with similar characteristics in terms of its feature values would be assigned to in the future. The mechanism that attract an object to other objects are the similarities of their statistical numerical fetuses to the clusters mean feature values. As new objects are added to the cluster the mean values of the cluster are adjusted accordingly. The algorithm iterates thorough the set of objects until the change of the mean features of the clusters is minimal under the given condition by the user.

$k$ -means clustering is a well known clustering algorithm where the number of clusters  $k$  is a number that comes from human knowledge and must then feed into the machine learning algorithm. The problem domain must be something well known so an appropriate  $k$  is chosen. The initial  $k$  clusters are then chosen usually at random from among the training data objects. The algorithm iterates through all the objects assigning each object to the closest clusters based on a similarity measurement. The similarity measure used in this research is the Euclidian distance from the clusters center to the object itself. According to Witten, et al. [39], a Euclidian distance measurement is sufficient in clustering if the features are of numerical values. The features are the dimensions of the objects and the cluster's center is represented as the mean of all the object within its cluster. The mean is adjusted after each iteration through the entire data set of objects. The clustering algorithm ends after the mean of all the clusters stabilizes. The criteria for stabilization is given usually by a threshold for the change in delta values of the mean object of the clusters. This step in finding the clusters is called the training step. Theses clusters developed from the training step is used as a classifier for future observed data objects. For this research the object

used for clustering are the process flow. Pre-processing the CMAT data, extracting the processes flows, and converting the flows into a CSV file as led to a file format that a machine learning clustering program can use as input. Cluster the flows and learn from the clusters is done by a program called Weka.

### 3.5 *Similarity Measure*

The measure of distance from one point  $x$  to another point  $y$  is the Euclidean distance and defined as:

$$d = ||x - y|| = \sqrt{\left(\sum_{i=1}^n (x_i - y_i)^2\right)}.$$

### 3.6 *Designator Analysis of Host-Based Process Flow Feature*

The following subsections describe the process of the data collection, the extracted features using a time window to capture “flows”. Each feature, their significance and the reason why the features were chosen for the attack classification model is described.

*3.6.1 Script Data, Controlled Experiments.* The importance of a controlled experiment is to control the applications running in the machine. A user controls when an application runs and ends. Also, different applications can run at the same time giving the control to the user on the amount of noise desired during the data collect. Isolating a processes and correlating it with a behavior that correlates to users task can only occur through a controlled experiment. The script used for the controlled experiment is listed in Appendix A. The script includes 2 computers 7\_4 and 7\_5 both running Windows 7 OS.

*3.6.1.1 Efficacy of Clusters on Individual Processes.* After calculating the process flows from the data collect of the CMAT feature files the next step is to group the instances of flows together.  $k$ -means clustering is used to cluster the

instances of the flows. Cluster analysis is done to determine if the flows in a cluster are all of the same type or it has the majority of the same application name. If a cluster has flows of notepad.exe processes then that cluster is labeled as the notepad.exe cluster and examined if the majority of all the notepad.exe is not spread out to many clusters but is being assigned to this cluster.

*3.6.1.2 Cluster Parameters.* The parameters for the model includes the number of clusters ( $k$ ) needed to accurately characterize application types of the collected raw feature files. There are two parameters that are unknown in the problem domain. A set of tests conducted to determine the best values for these parameters. Optimizing these parameters takes some domain knowledge of the data collect. A query to return the number of distinct application names is done as an initial  $k$  for  $k$ -means. That number is used as the amount of clusters for  $k$ -means.  $k$ -mean algorithm is executed on a given data set 10 times to see cluster stability and convergence. To check for stability of the clusters during the 10 runs of  $k$ -means with the three most populated clusters are compared from one run to the other observing the populations amount does not changes. If the population numbers are relatively the same this is a strong indication of cluster stability. Measuring each flow to all the clusters centers is made to verify that the clusters that  $k$ -means assigned are actually the closest in Euclidian distance.  $k$ -mean is executed twice more with  $k$  equals to twice the amount of distinct application names and  $k$  equals to half the amount of distinct application names. The amount of  $k$  is determine by two factors the clusters homogeneity of population application names and distance from other clusters. This is repeated for the sliding window sizes of 5, 10, and 15.

Three different window sizes to (5, 10, and 15) were selected. Because memory dumps are performed every 30 minutes, the window size must be small enough to isolate the anomaly and then be able to further investigation using the memory dumps close to the time of the suspected malicious activity. If the window size is too big an analyst must filter through multiple memory dumps.

*3.6.1.3 Anomaly Detection Threshold.* A data set of labeled known anomalous activity is evaluated using models generated previously from the script in Appendix A. The result of the assigned membership is analyzed to identify if the processes during these times differ and deviate from the center significantly. The further away from the assigned cluster is assumed to be anomalous. A threshold value is determined using density the cluster and where the known anomalous processes distance compares to the centroid of the assigned membership. To identify the anomaly detection threshold a rough fuzzy membership using the average distance of all the flows in the clusters plus the standard deviation is considered the threshold for each cluster.

*3.6.1.4 Data Collect and Problems.* The labeled data for normal activity was collected using two computers running the Windows 7 OS. The script in Appendix A lists all the tasks performed with the times of execution. The user tasks include opening, altering, and downloading files from internet, opening and closing user applications, browsing the internet, and using Microsoft office applications such as Word, Excel, and Outlook. The objective is to have a suitable representation of application types in the labeled data collect and user tasks.

The labeled data for malicious activity used Metasploit as the software tool to penetrate the computer traversing through the file system, killing processes, downloading files, and taking a snapshot of the desktop. The exploit used was hiding its process (PID = 3368, name = notepad.exe). The purpose of the malicious activity data collect is to observe these flows and measure where these flows would be assigned in the model and if they measure outside the threshold of the cluster.

*3.6.2 Uncontrolled Experiments.* The purpose of the uncontrolled experiments is to determine if a pattern of behavior of processes from different machines was performing similarly. An internet browser in one machine is performing similarly to an internet browser in another machine. The difference from the uncontrolled experiments are the type of applications are unknown. The behavior associated with

time could be correlated to the controlled data in the controlled experiments. The uncontrolled experiments the behavior cluster is unknown with only the application names giving the best guess on application behavior.

The training data acquired for the uncontrolled experiments was from the HACKFEST exercise held at the Air Force Institute Technology (AFIT), Wright Patterson Air Force Base OH, from September 11 , 2011 - September 12, 2011. The two day exercise consisted of two teams, the Blue Screens of Death (BSOD) and Ctrl-Alt-Elite(CAE). The objective of both teams was to defend their computer assets from attacks or intrusions and at the same time go on the offensive and attack the other team with any intrusion techniques. A point is given for each successful attack by each team. At the end of the two day exercise the team with the most points is the winner of the HACKFEST exercise.

HACKFEST is the capstone exercise for Air Force ROTC cadets from different universities in the United States participating in the Advanced Cyber in Engineering (ACE), Cyber Boot Camp. This is a specialize training for college ROTC cadets going into the field of Engineering and Cyber. It's a 10 week program teaching the techniques of cyber warfare and ends with the HACKFEST exercise that put their training into practice in a military cyber warfare setting.

Team BSOD had about 27 VMs it was responsible for defending and CAE about 30 VMs it was responsible for defending. Host log files for each VM were captured in a single file in a syslog format for the BSOD team. The host machine were running Xen with windows7 or windows 2003 virtual machines running on the host. A memory capture and situational memory forensic tool (CMAT) was running on the host Linux. CAE team had virtual machines part of their infrastructure and acted like separate physical computers. The host logs files were captured for all virtual machines. For each computer system and virtual machines a memory capture tool was used to take a snapshot of the memory approximately twice per minute. After the HACKFEST exercise it was identified that attacking machines were not dumped, and some VM's



were not up and running during the HACKFEST. Network traffic was captured for the CAE team while the BSOD network capture software crashed early with no network traffic data captured for BSOD. Paper logs were supposed to be used by both teams to record all major events of the exercise but both failed to record any events of the exercise.

During the HACKFEST exercise, Windows2k3 or Windows7 was the guest's operating system on the host Linux machines running CentOS with Xen as the Hypervisor. The ROTC cadets goals were to infiltrate the other teams VMs and at the same time defend their VMs. CMAT was running on all hosts machines and produced six text files containing the memory features in a tab-delimited. During the exercise CMAT was configured to dump the memory features twice every minute.

*3.6.2.1 Combined Baseline.* The flows of one computer was clustered to create models of normal behavior. To see if the validity of combining flows from different computers and arriving at a usable solution was also achieved flows from the different computer of the same OS in the HACKFEST exercise were combined and  $k$ -means was run against those flows. The purpose was to see if liked applications from one machine group together.

### ***3.7 Disadvantage of the Selected Research Methodology***

Flow generation and model creation was highly dependent on the CMAT data. Dealing with inconsistent data collect times was difficult and looking at the times of these file names that represents times of capture was mostly a manual processes to verify the frequency of the output files and events. At any point in time there were missing files and so the features of those files that some missing files were counted as zeros for that time of the observation. The system drivers was not associated with any processes but were associated with the time of the capture. The system drivers just counted the drivers for all the processes during the time window the system driver information was available. During the time the system driver information was not

available it was counted as zeros for the flows or the counts from the past 30 minutes from the last time the information was available.

Clustering using  $k$ -means might not have been the best way to cluster the flows. A density clustering like expectation maximization might have been more appropriate. With the current method  $k$ -means assumes that the clusters are of the same size clusters with similar densities. The distance of these clusters were close because the real changes in data by looking at the flows were the changes in the DLLs, ports if it interacted with the internet, and file permissions. The rest were mostly zeros. Looking at these lack of changes to the other features a feature selection mechanism could have been implemented to decrease the 194 dimensional space. This research process spent the majority of the time trying to find out how to get from the raw CMAT feature files to a processes flows. After a method was developed and executed to create flows the next step was creating flows from all these CMAT files. This was an overwhelming task resource wise, it was also easy to get lost in the data where it was almost too much information even when the abstraction was made from raw feature files to flows.

### **3.8 Summary**

The clustering of process flows and the approach discussed is meant to provide results that validate the concept of grouping like processes together is possible. These grouping of clusters started with an idea to use CMAT raw feature files in a usable way to detect anomalous processes. No gold standard of a data set was made using CMAT feature files and no work has been done using a feature set of processes in memory. A comparison of methods is not possible and accuracy measures cant really be obtained unless labeled data of flows is provided. Only a limited number of labeled data can be obtained of anomalous flows. The technology is simply not mature enough to provide this type of system. When the sensor technology is mature enough to provide the flexibility to quickly and efficiently provide flows in near real time only would this system be up and running. This chapter highlighted the research goals and hypothesis,

a description of creating the database for generating flows, the ideal test environment, and the cluster designator analysis along with the assumptions and limitations. The results of the limited testing could be found in Chapter 4. The results in chapter 4 does show that new flows from another machine was clearly closer in euclidian when measured in the same solution space to the same cluster with the same application name from another machine. Statically as described by the process flows these process are similar. Due to some technical issues not all planned tests were executed so future work must be included for more of an exhaustive verification of this proposed method.

## IV. Results Analysis

To determine if host derived flow based features could be used to behaviorally identify activities on a computer two data sets were generated. The first data set consisted of scripted events in which all of the user activities and the timing of those activities are known. The second makes use of an unknown situation and is a collection of data from a ACE HACKFEST defense exercise. Unfortunately, both of these data collections have flaws associated with data collection tool malfunctions. These flaws appear as gaps in the timeliness of collection, increasing delays, and missing data.

The following sections discuss the analysis of these two data collections. This analysis does show that the flow features do result on a strong behavioral clustering. However, because of the problems with the data, these results are exploratory and no definitive conclusions are drawn.

### 4.1 *Controlled Experiment*

Table 4.1: Timeline for Controlled Experiment.

Period	Window	Duration	Frequency $\frac{1}{cmaoutput}$ (sec)	Sensor Capture
1	1 - 40	2012 03 21; 11:33:13 - 2012 03 21; 11:32:36	30	40
2	41 - 57	2012 03 21; 11:53:14 - 2012 03 21; 12:02:37	35	17
3	60 - 85	2012 03 21; 12:24:39 - 2012 03 21; 12:33:41	22	26
4	86 - 116	2012 03 21; 12:39:05 - 2012 03 21; 12:43:40	9	31
5	117 - 159	2012 03 21; 13:52:46 - 2012 03 21; 14:32:06	56	43
6	160 - 166	2012 03 21; 14:57:49 - 2012 03 21; 15:01:49	40	7
7	167 - 189	2012 03 21; 15:28:16 - 2012 03 21; 15:41:54	37	23
8	190 - 200	2012 03 21; 16:05:17 - 2012 03 21; 16:12:07	47	11
9	204 - 221	2012 03 21; 16:49:30 - 2012 03 21; 17:01:57	44	18
10	222 - 229	2012 03 21; 17:26:28 - 2012 03 21; 17:32:02	48	8

The controlled experiment provides data to correlate and label behavior of user interaction to known applications. The test included opening internet browsers, surfing the internet to websites such as facebook.com, youtube.com and google.com. The labeled behavior included creating a word document using winword.exe saving the file and deleting the file. It also includes user interaction with the GUI of the Microsoft office products like word, excel, powerpoint and outlook, and traversal of the file system, altering files, moving files to different folders, and deleting files.

The goal was to capture the activities of a normal user on a personal computer. The drawback of this process is that there is no documented normal user behavior and normal activity a person would perform on a computer. A casual computer user might use an internet browser for most of the computer interaction for checking email, online banking such as paying bills, visiting social media websites such as youtube.com, facebook.com, or twitter.com, and creating word documents. Capturing a “normal user” on a computer has too many factors such as personal interest, age, work, and school. Appendix A shows the script of the actual user interaction with the computer systems.

Unfortunately the CMAT feature files did not capture all scripted behavior of the different application types. It did successfully capture the parts of the script which included sending email using outlook.exe, opening calc.exe, opening notepad.exe and using explore.exe. The rest was not captured because of sensor error and not knowing that it crashed. Table 4.1 shows the sensor captures for computer 7\_4. The frequency from time period 3 and 4 was too different that the flows created during these time are assumed invalid. Appendix A showed that two computers were part of the script but it was determined that computer 7\_5 was unusable for clustering as the frequencies were too sporadic. The focus is to demonstrate that the script data and the application in the data were clustering similar applications were to track applications present after time period 5. Those flows are considered valid as the frequencies show smaller variance than the earlier time periods. The four applications being analyzed and tracked were explorer.exe, calc.exe, outlook.exe and notepad.exe. Internet browser behavior were also analyzed with two different browsers, chrome.exe, and iexplore.exe

*4.1.1 Parameter Settings.* The parameters being optimized are the number of clusters  $k$  and the size of the sliding window. For this data set the sliding window of 5 discrete time intervals was chosen. As Table 4.1 shows that time period 6, 8 and 10 shows that during those time periods a sliding window of 10 or 15 would be too large of a window and therefore these periods would not be included for sliding window 10

and 15. To capture the behavior for these periods only a sliding window of 5 could be used. In the uncontrolled experiments where the time periods were much greater analysis of increasing the time windows observed. For the uncontrolled experiments the sliding window of 5 was the best choice.

The size of the sliding window is highly dependent on the sensor output on the raw feature files. Sometimes have a sliding window of 10 is too large because of the gaps in times. The sliding window used was a sliding window of 5 because it captured small enough increments so that gaps occur less than for 10 or 15 instances of a time event. Ideally, you want to have small enough sliding window to narrow in the times of the anomalous process but big enough that a user is not bombarded with data.

The structure extracted for clustering flows is a vector of 194 dimension long with only 191 being used for clustering. This system only needs cluster centers and threshold values to detect anomalies as outliers of the assigned clusters. These vector are the centers of these clusters, it is calculated as the average of all features or dimensions by all of its members of the cluster. The parameters therefore are the number of clusters to use in the learning phase to get the mean vector. This was calculated by starting with 62 clusters because there are 62 distinct application name within the training data. After running  $k$ -means 30 times, the model with the best sum of squared error was evaluated. During the evaluation it was discovered that flows from other clusters were closer to its center than clusters it was assigned to.  $k$ -mean ran again but doubling the  $k$  to 124. There was one empty cluster so the  $k$  used was 123. At this setting all cluster members are closer to their own center than any other center.

The threshold is the last parameters that needs to be determined. This is a user defined value based on observation of the cluster's population and cluster type. Some clusters might not have a threshold value due to the randomness of the type of processes that are contained in the cluster. Application type processes do cluster within one cluster containing the same application name. The threshold value used

was the maximum distance of an instance within the cluster. The calculator, note pad, and outlook processes did fall within the threshold value. The threshold value can be adjusted depending on the types of application the cluster is characterized as. An internet browser will have a larger threshold than notepad.exe because a browser accesses more services, resources, and network ports which affect the values of features in a flow. A standardize threshold value would not work for a clustering system to detect an anomalous process. The following discusses the Davies-Bouldin and the Dunn index as two ways to evaluate the relationship between members in the same clusters and relationships between other clusters.

*4.1.1.1 Davies-Bouldin index.* The Davies-Bouldin index [40] is calculated by

$$DB = \frac{1}{n} \sum_{i=1}^n \max_{i \neq j} \left( \frac{avgdist_i + avgdist_j}{d(c_i, c_j)} \right).$$

The  $avgdist_i$  and  $avgdist_j$  is the average distance of cluster  $i$  and  $j$  to its centroid.  $d(c_i, c_j)$  is the distance between the two centroid  $c_i$  and  $c_j$  and  $n$  is the number of clusters. The numerator represents the intra-cluster relationship and the denominator represents the inter-cluster relationship. With this index a smaller value is the best model because it represent dense clusters as well as cluster far apart from other clusters.

*4.1.1.2 Dunn index.* The Dunn index [41] is calculated by

$$D = \min_{1 \leq i \leq n} \left( \min_{1 \leq j \leq n, i \neq j} \left( \frac{d(i, j)}{\max_{1 \leq k \leq n} d'(k)} \right) \right).$$

The numerator is the distance between clusters  $i$  and  $j$  and  $d'(k)$  represents the intra-cluster distance of cluster  $k$ . Inter-cluster distance is the maximum distance between any pair of elements in cluster  $k$ . This index is different from the Davies-Bouldin that a larger value is considered the best model to select.

These two indexes assign the best scores to models that produce clusters that are dense, having high similarity measures and with clusters far away from each other. The problem with selecting the number of clusters with just the Davies-Bouldin and Dunn index criteria results in flows that are not nearest to its own named process center. The  $k$ -Means algorithm was iteratively doubled until the members of the clusters were actually the nearest to its centroid. The parameters that worked best for the controlled experiments was a sliding window of 5 and a  $k$  value of 121. Figure 4.1 and 4.2 are the graphs that show the Davies-Bouldin index and Dunn index for window 5 and window 10 as the number of cluster  $k$  increases. Figure 4.1 shows the ideal number of cluster  $k$  for a sliding window of 5 is around 40 using the Davies-Bouldin index and for the Dunn index the graph shows a value of approximately 25. Figure 4.2 shows similar results for the Davies-Bouldin but for the Dunn index shows a  $k$  value less than 10 is ideal. These might represent the best cluster using inter-cluster and intra-cluster relationship but measuring the actual flows to its assigned cluster does not result in the nearest cluster. This indicates the clusters are likely too close together and therefore having more clusters will provide better separation.

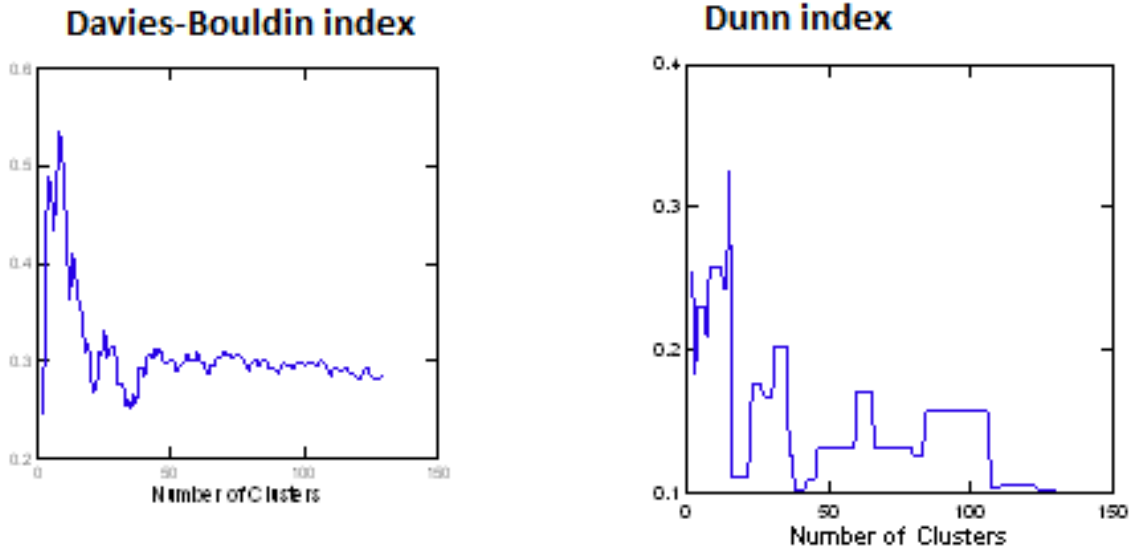


Figure 4.1: Index Score for Window 5.



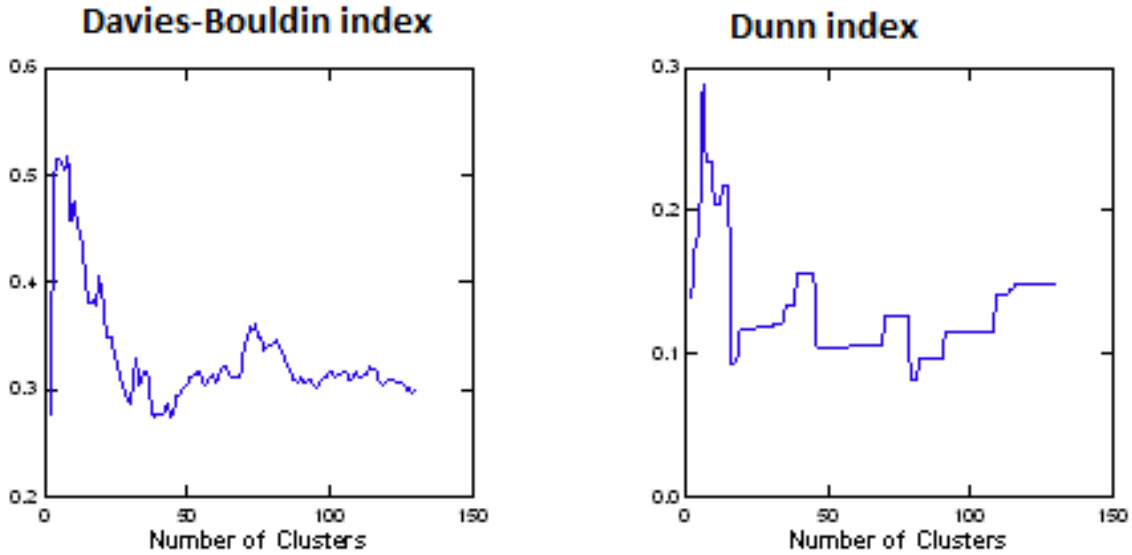


Figure 4.2: Index Score for Window 10.

*4.1.2 Data Efficacy of Clusters on Individual Processes.* Table 4.2 shows the 30 most populous  $k$ -means clusters from the normal behavior on computer name 7.4.  $k$ -means clustering algorithm executed 30 times to see cluster stability of the top three populated clusters. The one with the lowest sum of squared errors was evaluated on their cluster population to verify if the members of the clusters have the lowest Euclidian distance to its center. It was found with  $k=62$  with a sliding window of 5, that members of clusters were actually closer to other clusters than their own.

An observation that stands out is that one of the clusters contains 30 percent of the flows even if you change  $k$  from 62 to 124. The reason for this large number of flows is due to the nature of the processes flow with the SearchProtocol application. This processes is always present in the background due to the way the OS uses it. It may be running continuously in the background. It was observed that other processes accompany this flow. Whenever a flow with the application name SearchProtocol is present 15 other process flows are also present from the same time window. Calculating the Euclidian distance away for the groups mean vector for these 15 and the

Table 4.2: Computer 7\_4  $k$ -Means 3 results of the most populated clusters.

Iteration	Cluster 1	Cluster 2	Cluster 3	sum of squared errors
1	2840 (29)	612 (6)	564 (6)	1719.891285
2	2821 (29)	569 (6)	563 (6)	1554.308249
3	2827 (29)	643 (7)	563 (6)	1003.404694
4	2846 (29)	564 (6)	563 (6)	1162.532282
5	2846 (29)	645 (7)	572 (6)	1180.472643
6	2846 (29)	607 (6)	315 (3)	1130.284151
7	2846 (29)	617 (6)	455 (5)	2239.774589
8	3021 (31)	668 (7)	562 (6)	1788.961691
9	2827 (29)	573 (6)	315 (3)	2231.603146
10	2846 (29)	664 (7)	455 (5)	125.5029982
11	2846 (29)	570 (6)	565 (6)	1887.717425
12	2846 (29)	667 (7)	341 (3)	1431.417298
13	3002 (31)	564 (6)	563 (6)	1725.158949
14	2846 (29)	667 (7)	440 (4)	1370.507369
15	2827 (29)	563 (6)	315 (3)	1452.995254
16	2846 (29)	571 (6)	443 (3)	1126.979472
17	2846 (29)	656 (7)	465 (5)	1806.659177
18	2846 (29)	610 (6)	464 (5)	1621.125406
19	2846 (29)	534 (5)	408 (4)	1681.481552
20	2846 (29)	704 (7)	425 (4)	1026.585827
21	3009 (31)	621 (6)	419 (4)	1413.074045
22	3021 (31)	561 (6)	402 (4)	1747.645128
23	3021 (31)	612 (6)	562 (6)	1145.36198
24	2846 (29)	610 (6)	315 (3)	984.4036026
25	3021 (31)	667 (7)	453 (5)	1211.900995
26	2846 (29)	565 (6)	422 (4)	1514.965077
27	2827 (29)	621 (6)	437 (4)	1184.912736
28	2846 (29)	657 (7)	341 (3)	1499.906042
29	2846 (29)	536 (5)	406 (4)	1482.422749
30	3002 (31)	563 (6)	541 (6)	3000.298888

Table 4.3: SearchProtocol and Accompanying Processes.

896	SearchProtocol
1048579	NO NAME
1048586	NO NAME
1114120	NO NAME
1769477	NO NAME
1835016	NO NAME
2162692	NO NAME
5308422	NO NAME
5636100	NO NAME
6160386	NO NAME
7143427	NO NAME
7208963	NO NAME
7536644	NO NAME
7733251	NO NAME
1481769136	NO NAME
2233238792	NO NAME

process flow with the application name of SearchProtocol shows all are exactly the same distance away from the mean vector. SearchProtocol flow has accompanying flows wherever it was observed and implies that they are the same. Table 4.3 list the SearchProtocol PID = 896 and the other 15 PIDs that accompany it.

Table 4.4: Controlled Experiments Process Flows.

PID 164	notepad.exe	2276	calc.exe
WINDOW	CLUSTER	WINDOW	CLUSTER
190 – 222	15	190 – 222	15
PID 3588	outlook.exe	PID 2929	explore.exe
WINDOW	CLUSTER	WINDOW	CLUSTER
66	8	60 – 66	98
67 – 69	93	67 – 68	27
70	32	69	69
117	104	70	48
118 – 154	89	117–154	98
155	92	155	119
160 – 161	89	160 – 161	98
162	92	162 – 222	119
167 – 195	89		
196	92		
204 – 207	88		
208	32		
222	7		

*4.1.3 Tracking distinct processes.* Table 4.4 describes the four processes in the computer system focused on for data analysis. Any flows before window 118 are assumed invalid because 118 refers to time periods where the flows are invalid. Table 4.1 correlates the bad flows with the time periods and time windows of the data sets. Two processes behaving the same, notepad.exe and calc.exe have all their flows assigned to cluster 15. The script did not include any user interaction to these application so having both applications belong to the same cluster is likely the correct cluster assignment. This cluster might represent basic GUI behavior of both processes. Table 4.5 shows the average DLLs for cluster 15, 89 and 92. Cluster 89 and 92 are the clusters outlook.exe are assigned to. Comparing the counts of DLLs associated with cluster 89 and 92 shows greater DLL usage than cluster 15 as seen in Table 4.5. The file handles with read, write, delete and no permissions in Table 4.6 show a correlation that cluster 89 and 92 has greater number of file handles associated with the cluster. The only difference between 89 and 92 is the presence of registry key features as shown

in Table 4.7. The behavior that can be assigned to 89 and 92 is that these clusters contain flows that perform more file manipulation and cluster 15 might be the GUI of the application as calc.exe and notepad.exe have no related function beyond a GUI interface to the user.

Table 4.5: Controlled Experiments Process Flows with DLL Features.

CLUSTER	DLLST0	DLLST1	WindowDLLS	MINDLLS	MAXDLLS	AVGDLLS	STDDLLS	VARDLLS
15	27	27	135	27	27	27	0	0
89	149.0938	149.0938	745.4688	149.0938	149.0938	149.0938	0	0
92	117.375	117.375	586.875	117.375	117.375	117.375	0	0
CLUSTER	Q1DLLS	MEDDLLS	Q3DLLS	FFTDLLS	FFTDLLS2	FFTDLLS3	FFTDLLS4	FFTDLLS5
15	27	27	27	135	65.1838	65.1838	27	27
89	149.0938	149.0938	149.0938	745.4688	359.9442	359.9442	149.0938	149.0938
92	117.375	117.375	117.375	586.875	283.3683	283.3683	117.375	117.375

Table 4.6: Controlled Experiments Clusters File Handles with Permissions.

CLUSTER	READPERT0	READPERT1	WindowREADPER	MINREADPER	MAXREADPER	AVGREADPER	STDREADPER	VARREADPER
15	5.3071	5.3543	26.7008	5.2756	5.3622	5.2992	0.0157	0.0236
89	19.0938	18.7969	95.1719	18.7969	19.0938	19.0313	0.1094	0.8906
92	25	25	125	25	25	25	0	0
CLUSTER	Q1READPER	MEDREADPER	Q3READPER	FFTREAPER	FFTREAPER2	FFTREAPER3	FFTREAPER4	FFTREAPER5
15	5.3622	5.3622	5.3622	26.7008	12.9235	12.9235	5.3594	5.3342
89	19.0938	19.0938	19.0938	95.1719	46.1554	46.1554	19.1182	19.1182
92	25	25	25	125	60.3553	60.3553	25	25
CLUSTER	WRITEPERT0	WRITEPERT1	WindowWRITEPER	MINWRITEPER	MAXWRITEPER	AVGWRITEPER	STDWRITEPER	VARWRITEPER
15	2.2756	2.3071	11.4567	2.2756	2.3071	2.2756	0	0
89	15.5	15.3125	77.4063	15.25	15.5625	15.4375	0.0938	0.625
92	17.25	17.25	86.25	17.25	17.25	17.25	0	0
CLUSTER	Q1WRITEPER	MEDWRITEPER	Q3WRITEPER	FFTWRITEPER	FFTWRITEPER2	FFTWRITEPER3	FFTWRITEPER4	FFTWRITEPER5
15	2.2913	2.2913	2.2913	11.4567	5.5403	5.5403	2.3034	2.2988
89	15.5313	15.5313	15.5313	77.4063	37.5471	37.5471	15.568	15.5529
92	17.25	17.25	17.25	86.25	41.6452	41.6452	17.25	17.25
CLUSTER	DELETEPERT0	DELETEPERT1	WindowDELETEPER	MINDELETEPER	MAXDELETEPER	AVGDELETEPER	STDDELETEPER	VARDELETEPER
15	3.6142	3.6614	18.2362	3.5827	3.6693	3.6063	0.0157	0.0236
89	5.0938	5.0156	25.3906	5.0156	5.0938	5.0781	0.0313	0.0625
92	9.125	9.125	45.625	9.125	9.125	9.125	0	0
CLUSTER	Q1DELETEPER	MEDDELETEPER	Q3DELETEPER	FFTDELETEPER	FFTDELETEPER2	FFTDELETEPER3	FFTDELETEPER4	FFTDELETEPER5
15	0.0236	3.6693	3.6693	3.6693	8.84	8.84	3.6727	3.6522
89	0.0625	5.0938	5.0938	5.0938	12.3129	12.3129	5.1002	5.1002
92	0	9.125	9.125	9.125	22.0297	22.0297	9.125	9.125
CLUSTER	NOPERT0	NOPERT1	WindowNOPER	MINNOPER	MAXNOPER	AVGNOPER	STDNOPER	VARNOPER
15	2.0394	2.0394	10.1969	2.0394	2.0394	2.0394	0	0
89	16	15.75	79.75	15.75	16	15.9375	0.0938	0.625
92	17	16.5	84.25	16.5	17	16.75	0.125	0.25
CLUSTER	Q1NOPER	MEDNOPER	Q3NOPER	FFTNOPER	FFTNOPER2	FFTNOPER3	FFTNOPER4	FFTNOPER5
15	2.0394	2.0394	2.0394	10.1969	4.9235	4.9235	2.0394	2.0394
89	16	16	16	79.75	38.6771	38.6771	16.0206	16.0206
92	17	17	17	84.25	40.9002	40.9002	16.75	16.5138

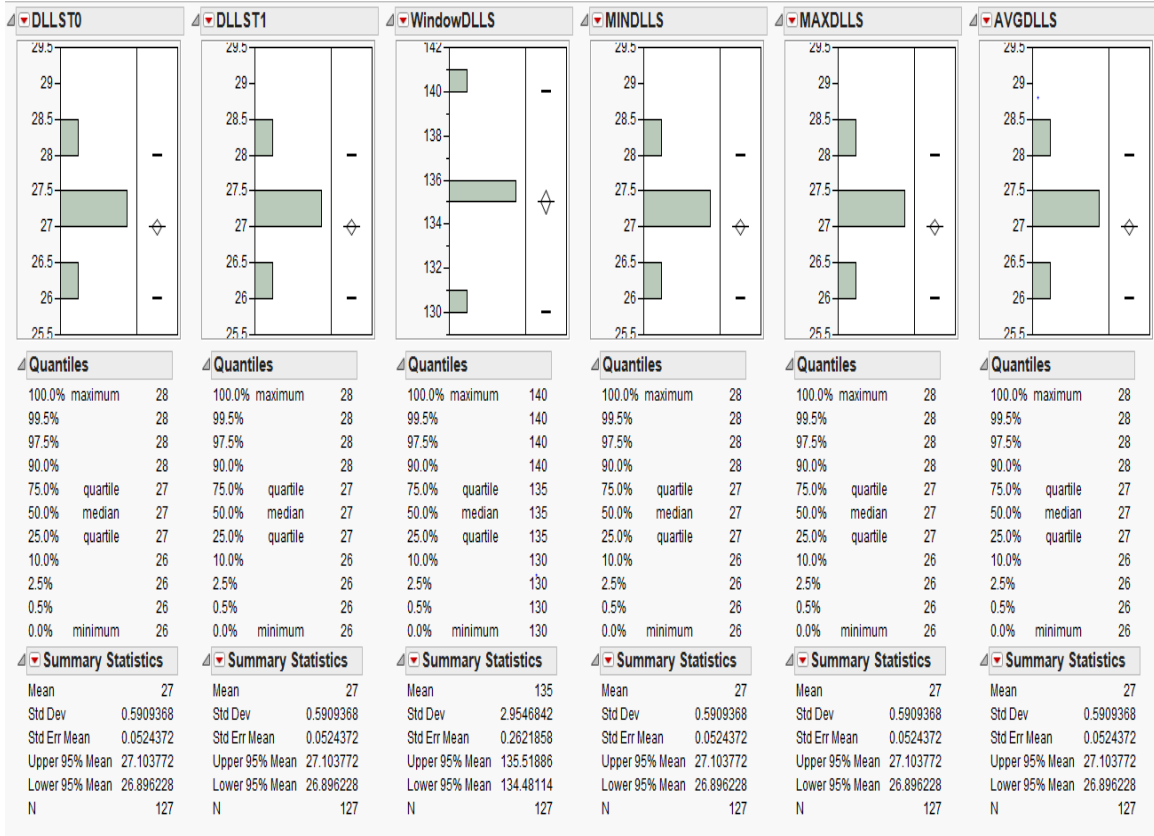
Table 4.7: Controlled Experiments Process Flows Registry Key Features.

CLUSTER	REGKEYST0	REGKEYST1	WindowREGKEYS	MINREGKEYS	MAXREGKEYS	AVGREGKEYS	STDREGKEYS	VARREGKEYS
15	0	0.5827	0.5827	0	0.5827	0.0709	0.2126	0.7402
89	0	0	0	0	0	0	0	0
92	0	52.5	52.5	0	52.5	10.125	20.75	468.875
CLUSTER	Q1REGKEYS	MEDREGKEYS	Q3REGKEYS	FFTREGKEYS	FFTREGKEYS2	FFTREGKEYS3	FFTREGKEYS4	FFTREGKEYS5
15	0	0	0	0.5827	0.5827	0.5827	0.5827	0.5827
89	0	0	0	0	0	0	0	0
92	0	0	0	52.5	52.5	52.5	52.5	52.5

The processes outlook.exe appears in window 66(12:26-12:29) to 70(12:29-12:31) in Table 4.4 but anything before 117 is considered an invalid flow. It re-appears in windows 117(12:43-13:54) to 222(17:01-17:28) at these times outlook.exe was opened at 15:34 and an email was sent. At time window 117 the processes flow was assigned

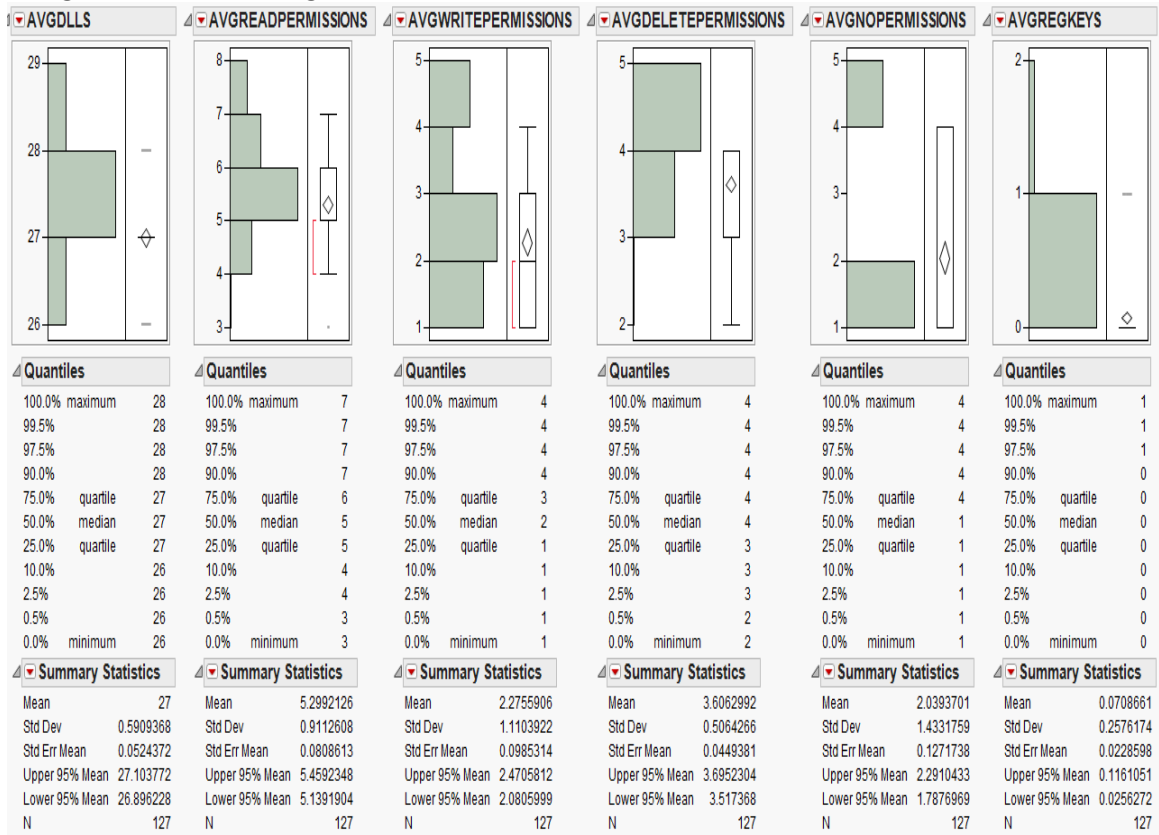
to cluster 104, this could be because of the opening of outlook.exe so the initialization of the application might start in cluster 104. The flows then jump to cluster 89 where 90 percent of the outlook.exe falls. The only time the outlook.exe changes cluster was 5 time units before a time gap, the sliding window was stopped and restarted after the time gap. Most of the flows did land at this cluster but the only user behavior for outlook that was captured was opening outlook and replying to an email message. More testing needs to be done to fully characterize outlook.exe for different permutations of sending email, attaching a file, downloading a file and clicking on a link. At the time of the data collect this was not evident that applications needs to have user interaction not just opening a file and letting it run. If an application has user interaction that needs to be captured in CMAT feature files.

Figure 4.3: DLL Features for Cluster 15.



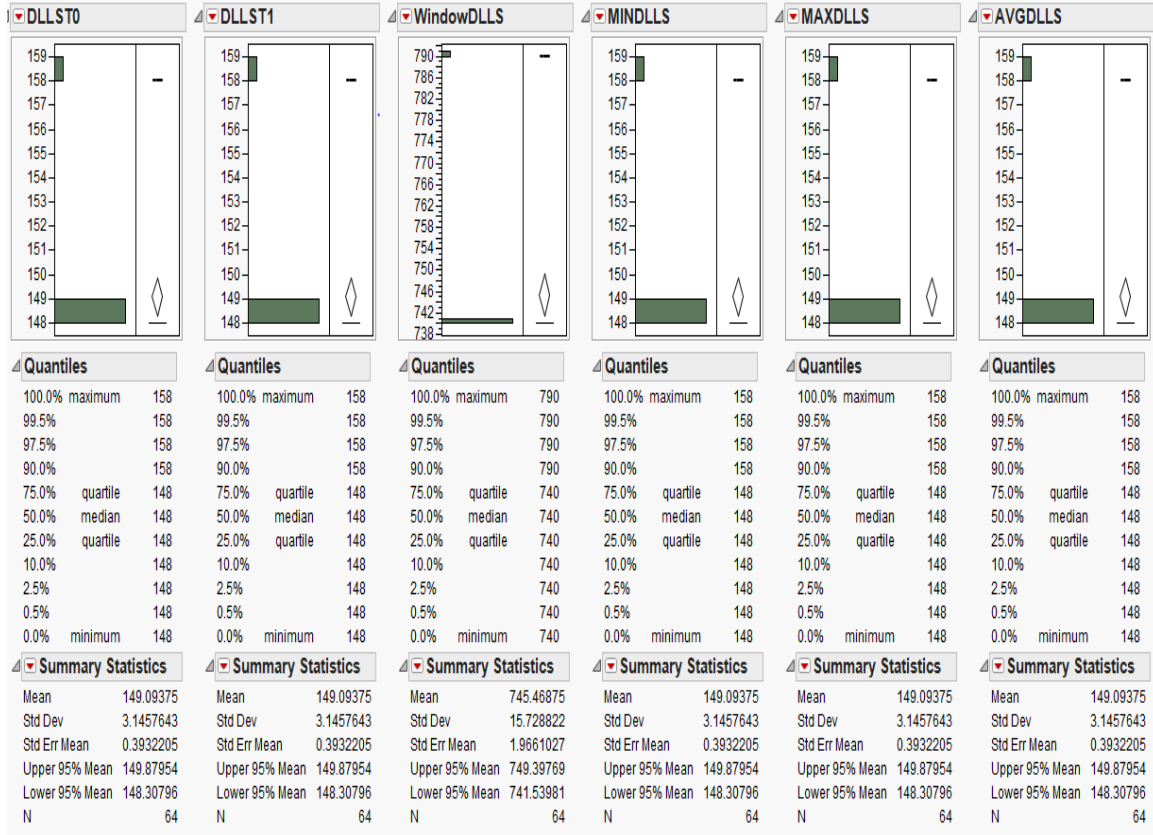
The process notepad.exe begins during time window 190 and stops at time window 222. During this time notepad.exe was opened but nothing was done to the

Figure 4.4: Average DLL and File Handle Permissions Features for Cluster 15.



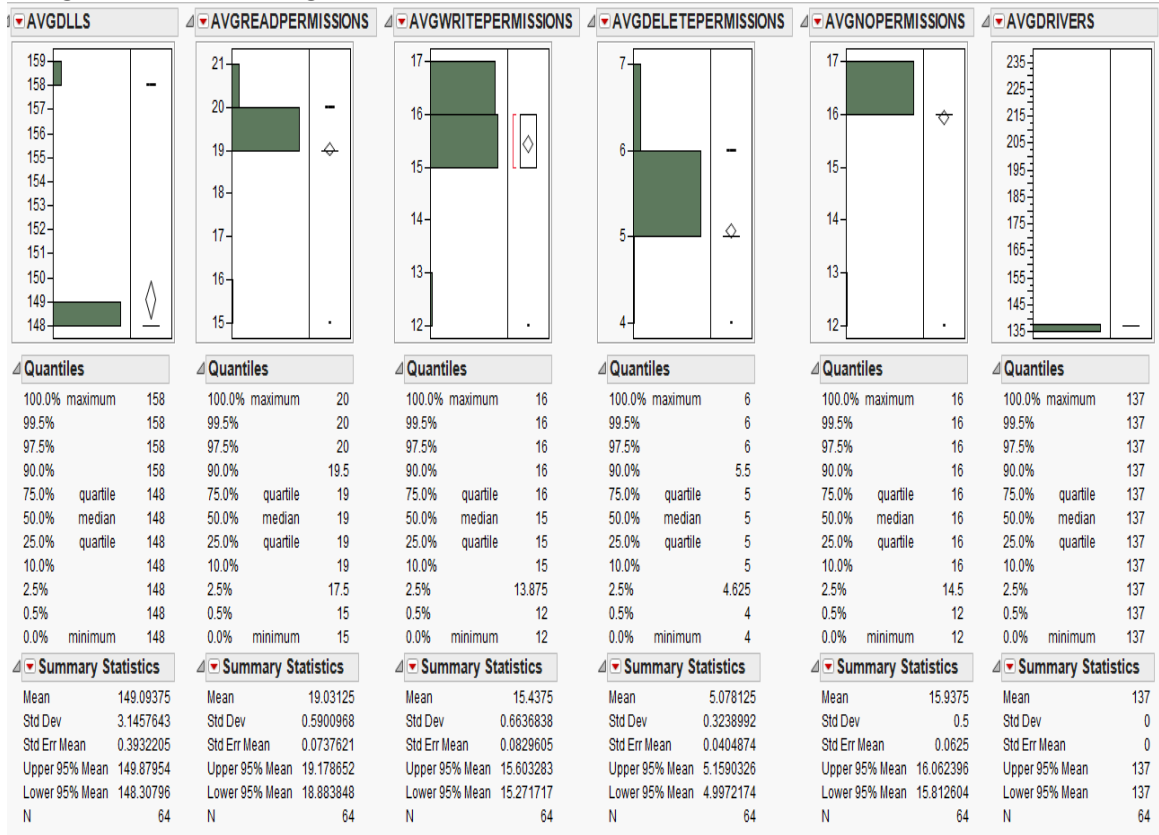
program. It remained open with no interaction or user input, no typing of text and saving a file. All flows are assigned to cluster 15.

Figure 4.5: DLL Features for Cluster 89.



The next processes of interest was calc.exe. The program started running at time window 190 (16:04) and the script confirms that this was opened at that time. From that time to the end of the data collect this program was running. All the flows were assigned to cluster 15. An observation was made that the calc.exe and the notepad.exe belonged to the same cluster. This was unexpected because the differences in the types of applications. An explanation for this was because of the lack of interaction to both processes. Statistically they are behaving the same in memory no inputs and no calculation were made to the calculator. No typing of text was made to the notepad.exe application. For the most part the same application name clustered together such as outlook.exe, notepad.exe and calc.exe. The last process type is explorer.exe, it was found in a few cluster but most of the cluster

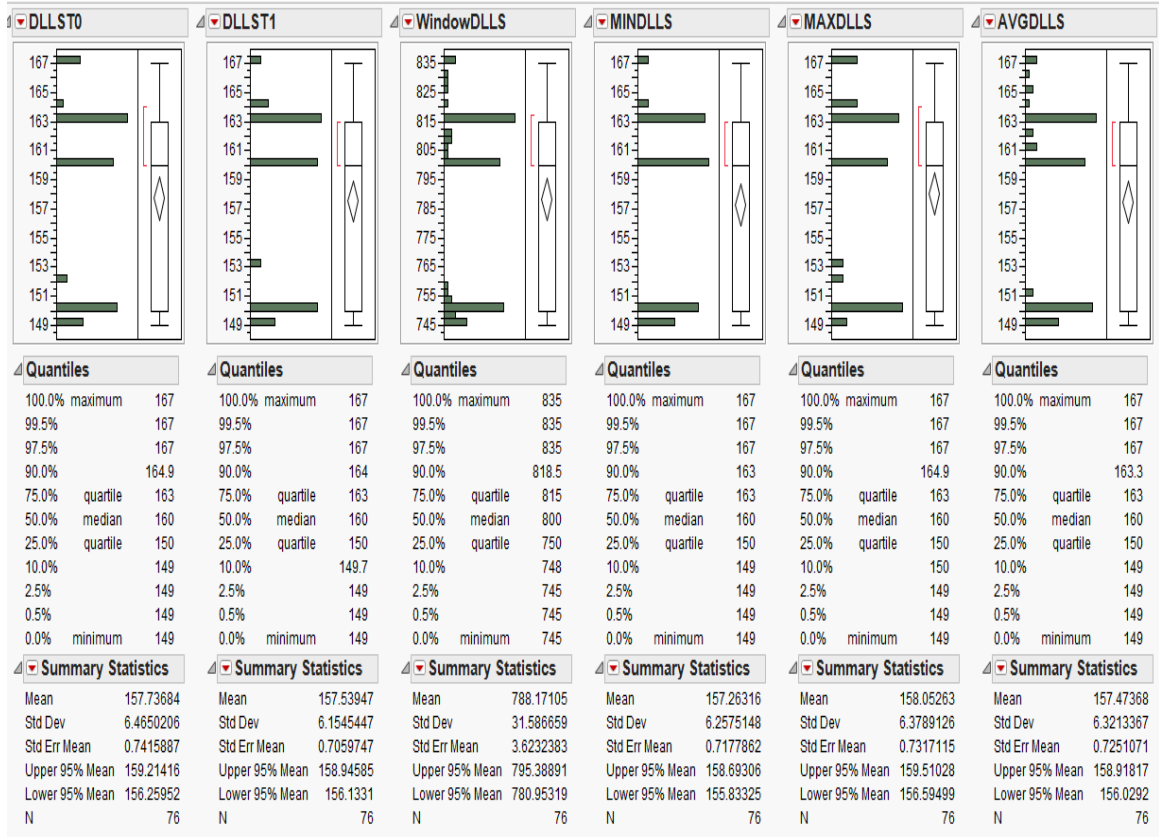
Figure 4.6: Average DLL and File Handle Permissions Features for Cluster 89.





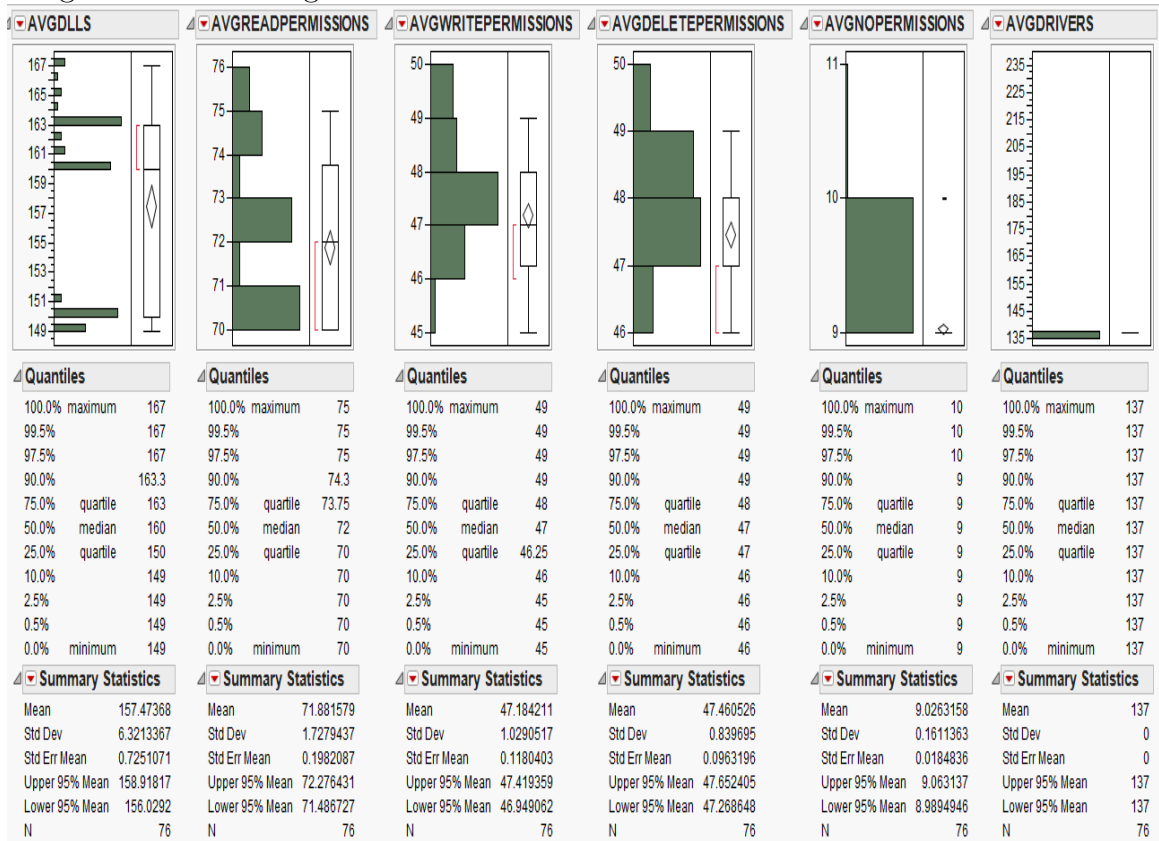
assignment was in cluster 98. Figure 4.3 and Figure 4.4 show the statistics for DLLs, and file handles with permissions of the features associated with cluster 15. Figure 4.5 and 4.6 shows the same statistics for cluster 89, and Figure 4.7 and Figure 4.8 show the same statistics for cluster 98. From the figures and the cluster features shows that cluster 98 has greater number of DLLs and file handles than cluster 15 and 92. This confirms that the process flows of explore.exe assigned to cluster 98 has more activity with respect to file handles and DLLs. Therefore more file manipulation activity is associated with cluster 98.

Figure 4.7: DLL Features for Cluster 98.



**4.1.4 Test of Clusters.** The main purpose for this research was to see if the novel way of representing processes in memory as flows could be used as a way to possibly detect anomalies. The first step is to determine if flows of the same application name are actually alike from the same machine or from a different machine.

Figure 4.8: Average DLL and File Handle Permissions Features for Cluster 98.



$k$ -means clustering was used to group similar instances together from one machine and flows from another machine are compared to verify that the clusters remain consistent. For example, the flows for calc.exe on one machine should be closest to the cluster that contains calc.exe when coming up with the model.  $k$ -means is executed on a training set then new unseen flows are measured to confirm that the predicted cluster was selected.

The scope was limited to several processes that are observed in both the test and training set for a better understanding and not to get overwhelm on the amount of data. Testing showed a flow with the application name running in one machine will select the correct cluster for the same application name from another machine. These flows are statistically the same with some variance. The exceptions to this were Windows operating system processes like svchost.exe that is responsible for a variety of services. A number of services share this in order to reduce resources. This was too difficult to narrow down into a few clusters because of the nature of this process and the numerous instances of it running.

A new set of flows previously unseen was measured against all the centers mean vector. This was to verify that the closest Euclidian distance of the same application type of the unseen flows are within the same cluster threshold defined as the maximum distance of instances during the training of the clusters.

Table 4.8: Timeline for Controlled Experiments with Anomalous Processes.

Period	Window	Duration	Frequency $\frac{1}{\text{cmatoutput}}$ (sec)	Sensor Capture
1	1 - 18	2012 03 26; 14:01:01 - 2012 03 26; 14:06:12	18	18
2	19 - 34	2012 03 26; 14:45:36 - 2012 03 26; 14:54:55	37	16
3	35 - 39	2012 03 26; 15:12:53 - 2012 03 26; 15:14:50	29	5
4	40 - 48	2012 03 26; 16:02:01 - 2012 03 26; 16:11:17	59	9
5	49 - 66	2012 03 26; 16:49:29 - 2012 03 26; 17:07:04	66	29

*4.1.5 Data Collect and Problems.* Table 4.8 presents the timeline for the data capture where the anomalous activity was captured. The data collect was inconsistent and yields no useful flows from feature files. The frequencies of the CMAT sensor did not stabilize. In Table 4.8 the frequencies of the feature files steadily in-

creased starting with 18 seconds during the first time period to growing three times in value during the fifth time period. The flows of application name of calc.exe, outlook.exe, and notepad.exe were extracted but only using the three largest frequencies of 37, 59 and 66. The predicted cluster of PID = 2628 of a unseen processes flow of calc.exe was 15. The distance of the new calc.exe to the cluster 15 was 27.6. There were 41 flows with the new extraction and 19 out of the 41 did accurately predict the new flows of calc.exe. Some of the reason for the misclassification were time gaps in the sensor outputs. The predicted cluster for the new previously unseen PID = 2876 notepad.exe was 15. 28 out of 47 accurately predict the new flows for notepad.exe. In Table 4.9 is the distance of the predicted distances for the anomalous processes hiding under the application name of notepad.exe. The distance are outside the threshold of cluster 15 but closer to other clusters. The predicted cluster assignment for the new previously unseen flow for outlook.exe performed poorly as 6 out 32 was closest to cluster 89. The behavior of outlook.exe during these time might have moved due to behavior differences from the captured model like user interaction.

Table 4.9: The anomalous processes hiding under notepad.exe

PID	cluster15	PID	cluster15
1008	869.5562	3368	869.4033
1008	651.6951	3368	651.5577
1008	419.1466	3368	418.2496
1008	262.7405	3368	260.0134
1008	138.2578	3368	129.0791
1008	142.7257	3368	129.0791
1008	142.7257	3368	129.0791
1008	142.7257	3368	129.0791
1008	264.5862	3368	260.0155
1008	419.6903	3368	418.2509
1008	651.8482	3368	651.5585
1008	869.3189	3368	869.404

The four new unseen process flows from another data set selecting the predicted cluster from the training set gives some indication that processes with the same application type from one machine are similar to another machine. This controlled

test was flawed and verification was not accomplished because the training data had flaws. The new unseen flows were considered invalid because the CMAT sensor was inconsistent with feature file outputs. The next sections discuss the uncontrolled experiments. These experiments were uncontrolled but the data and flows generated from this data set is considered valid. The frequencies are similar and the period of time is greater. A cluster behavioral analysis is presented using the uncontrolled experiments in the following sections.

#### ***4.2 Unlabeled Data Collect: Uncontrolled Experiment***

The purpose of the uncontrolled experiments is to determine if a pattern of behavior of processes from different machines was performing similarly. The clusters are unknown with only the application names giving the best guess on application behavior. The test seek to determine if a model can be extracted if the activity of the system is unknown.

The HACKFEST data was consistent for Windows 2003 OS but lacked the consistency of sensor output frequency in most of the Windows 7 machines. Only one Windows 7 machine was used for training and two machines were used to model Windows 2003 OS. Three machines were selected because of the regular frequencies they provided. The three machines were also chosen because of attainability of creating process flows. The method for generating flows was a time consuming processes because of the combinations of parsing the CMAT feature files, populating the MySQL database with the features, then querying for relevant statistical characteristics of a process. A trade off in the amount of training data and the amount of the test set used was made to focus on providing a method for using the flows generated. The purpose of the unlabeled data is to verify that within this type of data set application of the same type do cluster.

*4.2.1 Window 2003 OS, CMAT Captures.* The timeline for a Windows 2003 OS used is in Table 4.10. This data set had potential for analysis but the lack of

Table 4.10: Hackfest BSOD-02 TimeLine.

Period	Window	Duration	Frequency $\frac{1}{c_{matoutput}}$ (sec)	Sensor Captures
1	1 - 79	2011 08 11; 11:34:38 - 2011 08 11; 12:04:19	23	79
2	80 - 144	2011 08 11; 12:09:49 - 2011 08 11; 12:34:28	23	65
3	145 - 165	08 12; 07:29:07 - 2011 08 12; 07:36:22	21	21

different applications of the same type running on the machine limited its usefulness. There was only one internet browser and the rest of the applications were native to the operating system. The following subsections outline the types of applications running in a server environment with the focus emphasized in the next section that demonstrate the novel way of modeling behavior with clusters in the Windows 7 OS.

The number of process flows generated from computer BSOD-02 are:

- 3775 flows using a sliding window of 10 discrete time observations
- 2945 flows using a sliding window of 20 discrete time observations
- 1357 flows using a sliding window of 30 discrete time observations.

Table 4.11: Assigned PIDs Application Name.

Application	PID	Application	PID	Application	PID	Application	PID
Idle	0	cmd.exe	1332	svchost.exe	652	nc.exe	2116
System	0	svchost.exe	1340	svchost.exe	656	rundll32.exe	2152
smss.exe	4	SnareCore.exe	1360	svchost.exe	668	explorer.exe	2164
smss.exe	272	SnareCore.exe	1364	svchost.exe	720	fakefmt.exe	2248
IEXPLORE.EXE	284	SnareCore.exe	1368	svchost.exe	724	IEXPLORE.EXE	2356
csrss.exe	296	svchost.exe	1520	davcddata.exe	736	svchost.exe	2416
davcddata.exe	320	svchost.exe	1564	svchost.exe	740	cmd.exe	2488
csrss.exe	328	cmd.exe	1576	svchost.exe	756	wmiprvse.exe	2536
winlogon.exe	332	svchost.exe	1660	svchost.exe	760	wmiprvse.exe	2920
winlogon.exe	344	svchost.exe	1668	svchost.exe	776	IEXPLORE.EXE	3004
services.exe	356	svchost.exe	1732	svchost.exe	792	mmc.exe	3116
lsass.exe	392	msconfig.exe	1740	spoolsv.exe	912	wmiprvse.exe	3264
services.exe	404	alg.exe	1756	spoolsv.exe	928	cmd.exe	3372
lsass.exe	404	alg.exe	1768	msdtc.exe	936	logon.scr	3608
svchost.exe	416	alg.exe	1780	msdtc.exe	960	inetinfo.exe	3852
svchost.exe	564	dllhost.exe	1876	rundll32.exe	988		524767
nc.exe	568	explorer.exe	1976	svchost.exe	1056		2156518208
svchost.exe	584	explorer.exe	1988	svchost.exe	1064		3774876801
wmiprvse.exe	612	svchost.exe	2000	svchost.exe	1120		3779113917
cscript.exe	624	nc.exe	2028	inetinfo.exe	1144		3782974786
	628	cmd.exe	2060	inetinfo.exe	1192		
				cmd.exe	1208		
				svchost.exe	1316		

Table 4.12: Cluster Characteristics model for 16 clusters.

CLUSTER	AVG	MAX	MIN	Counts	STD DEV
1	56.64347	321.2886	33.61744	150	78.03961
2	11.38245	110.1848	5.983956	133	22.90466
3	303.7498	620.9369	117.1756	497	185.8206
4	1.13E-06	1.13E-06	1.13E-06	12	0
5	112.8024	112.8024	112.8024	252	3.40E-08
6	193.3992	374.7108	36.30457	1733	100.7969
7	125.7043	353.7071	14.51939	26	99.11451
8	298.8698	736.5652	43.63628	178	192.8164
9	2.558844	14.71335	1.401272	138	3.750963
10	6.384249	52.61472	3.400962	134	8.351216
11	46.75154	1090.828	22.17669	141	153.9762
12	63.10922	841.6387	28.92818	151	102.7512
13	321.9321	633.1781	153.3658	44	126.5231
14	2.678223	168.728	1.349824	126	14.85195
15	5.58E-07	5.58E-07	5.58E-07	56	8.47E-22
16	1.27E-06	1.27E-06	1.27E-06	4	0

Table 4.11 lists all the processes running on the system. The application types running on a server environment has small diversity. Most of the processes are related to the inner workings of the operating system. An assumption in the training data captured was that the majority of the processes reflects normal process and outnumbers potentially anomalous processes. In an ideal world, training data is clean and only contain non-malicious processes. All processes are assumed to be the correct processes as it identifies itself in the CMAT feature files. The problem is Processes that have to do with the operating system cant be contained in a few clusters.

### 4.3 Windows7 Operating System HACKFEST

Table 4.13: Hackfest BSOD-19 TimeLine

Period	Window	Duration	Frequency $\frac{1}{cmatoutput}$ (sec)	Sensor Capture
1	5 - 27	2011 08 25; 11:34:00 - 2011 08 25; 11:43:20	24	24
2	28 - 56	2011 08 25; 12:08:35 - 2011 08 25; 12:32:05	50	29
3	57 - 83	2011 08 25; 13:13:30 - 2011 08 25; 13:33:39	46	27
4	85 - 94	2011 08 26; 01:30:46 - 2011 08 26; 01:33:49	20	10
5	95 - 150	2011 08 26; 01:41:31 - 2011 08 26; 02:00:11	20	56

4.3.1 Cluster Analysis For Windows 7 OS: BSOD - 19. Table 4.13 shows the observation periods, the CMAT feature files output frequencies with the number

of file captures. The observation period 2 and 3 and the frequencies of the CMAT feature file outputs was approximately double the amount compared to the other period of observations (1,4, and 5). This type of inconsistency was common in the operation of the sensor outputs. Too many factors in the sensor environment plus the sensor was operating in virtual machine environment led to inconsistency of the raw feature outputs. The only way to verify that the feature files were outputting with a consistent frequency was after the data collect was accomplished then analyzing the file names that represents the time are equally spaced time intervals. No real time analysis was possible to verify the output files are generated on consistent intervals.

The following two data sets from the HACKFEST exercise computers BSOD-19 and CAE-18 were chosen as the best candidate for analysis. They displayed the most consistent frequencies with minimal changes from one period to another. The following sections describe some processes from BSOD - 19 and follow the behavior as the process progresses through time. The process flows calculated from the raw flow features are clustered using the  $k$ -means clustering algorithm. The sliding window size was 5, and the value of  $k$  was 121. The behavior process flows are characterized as the cluster assignment it is assigned to. The changes in cluster assignment are analyzed for differences in the features of the cluster. Membership analysis was done, the features of the clusters with key distinct difference was labeled or categorized describing the behavior.

#### 4.3.1.1 *Application Name = chrome.exe, PID = 3012, 3428, 1960, 2940.*

The process flows in Table 4.16 shows that there is a pattern of an initialization sequence of behavior. The cluster sequence starts with 67 then transitions to 79 and 109. After cluster 109 the behavior of the three chrome processes do not converge to a single cluster instead they exhibits different behavior as result of different cluster memberships.

Looking at the centroid centers of cluster 67, 79, and 109 shows that 79 has actual values for the features for port, UDP, and registry information. PortsT0 is the



Table 4.14: Hackfest BSOD-19 Applications.

<b>application</b>	<b>application</b>
chrome.exe	rubyw.exe
conhost.exe	rundll32.exe
Console.exe	SearchFilterHo
csrss.exe	SearchIndexer.
dwm.exe	SearchProtocol
explorer.exe	services.exe
GoogleUpdate.e	slui.exe
Idle	smss.exe
iexplore.exe	spoolsv.exe
javaw.exe	sppsvc.exe
LogonUI.exe	svchost.exe
lsass.exe	System
lsm.exe	taskeng.exe
mscorsvw.exe	taskhost.exe
msiexec.exe	TrustedInstall
nessusd.exe	VSSVC.exe
nessus-service	wininit.exe
nessusvrmanag	winlogon.exe
pg_ctl.exe	WmiPrvSE.exe
postgres.exe	wuaclt.exe
ruby.exe	

counts of ports at the beginning of the sliding window and its greater than 0. The same goes to the maximum ports (MAXPorts), and the Discrete Fast Fourier Transform of the magnitude of the greatest five discrete number of ports in the sliding window (FFTPorts, FFTPports2, FFTPports3, FFTPports4, and FFTPports5). There are ports associated with this process flow during the start of the processes.

Table 4.15: Hackfest BSOD-19 chrome.exe.

PID 1960		PID 3012		PID 3428		PID2940	
WINDOW	CLUSTER	WINDOW	CLUSTER	WINDOW	CLUSTER	WINDOW	CLUSTER
60	79						
61 – 62	109						
63 – 79	57						
80 – 81	51	81	67	81	67	81	67
82	68	82	79	82	79	82	79
83	119	83 – 84	109	83 – 84	109	83 – 84	109
		85 –	88	85 –	57	85 –	21
		–	85	–	57	–	85
		-145	85	-145	57	-145	21
		146	98	146	98	146	98

Table 4.16: Hackfest BSOD-19 Initialization, Ports.

cluster	PortsT0	PortsT1	WindowPorts	MINPorts	MAXPorts	AVGPorts	STDPorts	VARPorts
67	0	0	0	0	0	0	0	0
79	0	0.0877	0.2982	0.0877	0.2105	0.1404	0.0526	0.0526
109	0	0	0	0	0	0	0	0
	Q1Ports	MEDPorts	Q3Ports	FFTPorts	FFTPorts2	FFTPorts3	FFTPorts4	FFTPorts5
67	0	0	0	0	0	0	0	0
79	0	0.1491	0	0.2982	0.2849	0.2849	0.2469	0.2469
109	0	0	0	0	0	0	0	0

After the initialization sequence of the process flows to clusters 67, 79, then 109 for processes 3012 and 3428, their associated flows are assigned to 85 and 57 shown in Table 4.15. The differences of the features of cluster 85 and 57 are the counts of file handles for cluster 85 is slightly greater. On average its only greater by one unit. This might indicate that the behavior is the same and might be a good candidate to combine these clusters together. This research did not look into optimizing the clusters but is left for future research. This research focused on the creation of the flows and providing a proof of concept that these flows can characterize behavior in memory. Cluster 79 has the behavior of some type of communication over the internet. Cluster 57 and 85 has the behavior of some file processing or browser interaction without communication in terms opening ports.

In Table 4.15 the process flows with application name chrome.exe and PID 1960, after entering cluster 57 from window 63 to 79 shows a jump in cluster starting at window 80 to 81. This jump could be affected by the system as the initialization sequence of the other three processes starting at the same time window. Analyzing the feature averages for cluster 51 as represented as the mean of all the members associated for this cluster shows that cluster 51 was similar to 85 with no ports, TCP, and UDP features. The DLLs are greater than cluster 57 and 85. The file permission features for write, read, and delete in cluster 51 was less than in clusters 57 and 85. The chrome.exe process flows with PID 2940 shown in table 4.15 has a change in behavior during windows 85 to 89 and windows 95 to 145 jumping to cluster 21. PID 2940 and 3012 has similar behavior with differences in cluster assignment for 2940 from 85 to 21 during the time windows of 85 to 145. The only difference was in

cluster 21 in which no registry key features were present. The behavior was almost identical with a slight edge in the number of file handles was greater in cluster 21 on average.

Table 4.17: Hackfest BSOD-19 chrome.exe, iexplore.exe and nessus.

PID 188	chrome.exe	PID 3668	iexplore.exe	PID 3528	nessus-service	PID 3468	nessussvrmanag	PID 3548	nessus-service
WINDOW	CLUSTER	WINDOW	CLUSTER	WINDOW	CLUSTER	WINDOW	CLUSTER	WINDOW	CLUSTER
1	79	1	79	1	79	1	79	1	59
2 – 3	109	2 – 3	109	2 – 3	101	2 – 3	77	2 – 3	59
4 –	41	4 –	41	4 –	44	4 –	92	4 –	59
– 13	41	– 13	41	– 13	44	– 13	92	5 – 13	76
14	89	14	89	14	89	14	89	14	89
15 – 17	16	15 – 17	16	15 – 17	31	15 – 18	103	15 – 18	103
18	95	18	95	18	95	– 18	103	– 18	103
19	22	19	22	19	22	19	22	19	22
20 –	41	20 –	21	20 –	44	20 – 24	92	20 – 54	76
	41		21		44	25 – 78	15	55 –	108
– 79	41	– 79	21	– 79	44	79 –	90		108
80	65	80	51	80	63		90		108
81	51	81	51	81	63	– 81	90	– 81	108
82	68	82	68	82	68	82	110	82	68
83	119	83	119	83	119	83	110	83	119

4.3.1.2 *chrome.exe, iexplore.exe and nessus, PID = 188, 3668, 3528, 3468, and 3548.* Table 4.17 shows the relationships between process flows of the application name chrome.exe, iexplore.exe, nessus-service and nessussvrmanag. There are some underlying relationships between the flows and the time windows. For PID 188 and 3668 two different internet browser one chrome.exe and the other iexplore.exe shows an initialization sequence of behavior starting with cluster 79, jumping to cluster 109 for 2 time window periods, jumping to cluster 41 for 10 time window periods, jumping to cluster 89 for one time window, going cluster 16 for the next 2 time windows, then to cluster 95 and cluster 22. The two browser changes to different clusters after the initialization sequence. The processes has an ending pattern termination sequence of cluster 51, 68, then 119.

In Table 4.16 shows the initialization assigned clusters for chome.exe and iexplore.exe. Cluster 79 has the behavior of ports, TCP, and UDP activity assigned to the cluster as the numerical averages for the other clusters were 0. Cluster 79 may be user defined and labeled as network port activity as the behavior. The behavioral change between PID 188 and 3668 start at windows 20 to 79 where chrome.exe flows were in cluster 41 and the iexplore.exe flows were in cluster 21. The differences be-

Table 4.18: Hackfest BSOD-19 chrome.exe, iexplore.exe

Cluster	PortsT0	PortsT1	WindowPorts	MINPorts	MAXPorts	AVGPorts	STDPorts	VARPorts
79	0	0.0877	0.2982	0.0877	0.2105	0.1404	0.0526	0.0526
109, 41, 89, 16, 95, 22	0	0	0	0	0	0	0	0
	Q1Ports	MEDPorts	Q3Ports	FFTPorts	FFTPorts2	FFTPorts3	FFTPorts4	FFTPorts5
79	0	0.1491	0	0.2982	0.2849	0.2849	0.2469	0.2469
109, 41, 89, 16, 95, 22	0	0	0	0	0	0	0	0
	TCPT0	TCPT1	WindowTCP	MINTCP	MAXTCP	AVGTCP	STDTCP	VARTCP
79	0	0.0702	0.1754	0.0702	0.1053	0.0702	0	0
109, 41, 89, 16, 95, 22	0	0	0	0	0	0	0	0
	Q1TCP	MEDTCP	Q3TCP	FFTCPT	FFTCPT2	FFTCPT3	FFTCPT4	FFTCPT5
79	0	0.0877	0	0.1754	0.1648	0.1648	0.1343	0.1343
109, 41, 89, 16, 95, 22	0	0	0	0	0	0	0	0

tween these two clusters was the amount of DLLs for 41 was slightly greater with a difference of 10. The read, write and delete permissions of files associated with the cluster 41 was slightly greater but the feature that stood out for cluster 41 was it had registry key associated with it. Taking a look at the features for the termination sequence of 51, 68 then 119 shows that 51 had more DLLs than 68 and 119 with 119 having the least DLLs for that cluster. The files associated with those cluster show that 51 has more file handle permissions for read, write and delete, with cluster 68 having the second most and 119 had the least.

An interesting observation in Table 4.18 is the relationship between the two browsers chrome.exe, iexplore.exe and the nessus applications. Nessus is a software for vulnerability scanning of a target system. The purpose of the software is to find vulnerabilities on a target system by scanning the network, and the ports the target system resides. The behavior of this application deals with communication. The features of the 194 dimension of the processes flows has ports, TCP, and UDP information to capture changes in communication activity. Cluster 79 has port, TCP, and UDP activity shown in Table 4.16 and both nessus applications with PID 3528 and 3468 has the same start up cluster of 79. The behavior of nessus-service with PID 3528 had similar behavior of the entire life of the process to chrome.exe and iexplore.exe. The change was in window 15 to 17 it was assigned to cluster 31 instead of 16, from window 20 to 79 it was assigned to cluster 44 instead of cluster 41 or 21 and in window 80 to 81 it was assigned to cluster 63. The difference between

cluster 31 and 16 during the time window of 15 to 17 was cluster 16 had more DLLs on average about 6 times greater than cluster 31 and the file permissions was also greater. This is indications that cluster 16 was more active of file activity with both cluster having no port, TCP and UDP activity.

The two nesses applications with PID 3468 and 3548 have similar behavior during time window of 14 to 19 but from window 20 to 54 it is assigned to cluster 76 then cluster 108 from time window 55 to 81. These two clusters are almost the same with slight differences in the permissions features. Comparing these two cluster to the cluster in PID 3468 and 3528 of the same time window show that it has fewer DLLs associated with this cluster but has an increase in the file permissions associated with the two clusters. The applications with similar behavior are assigned to the same clusters. Nessus that uses ports, TCP and UDP activity shows that the process flows and the behavior as defined clusters similarly to the browser process flows.

Table 4.19: Hackfest BSOD-19 chrome.exe, iexplore.exe initialization cluster.

PID 616	chrome.exe	PID 2976	chrome.exe
WINDOW	CLUSTER	WINDOW	CLUSTER
1 - 2	56	1 - 3	49
3	113		
PID 2280	chrome.exe	PID 3380	iexplore.exe
WINDOW	CLUSTER	WINDOW	CLUSTER
1 - 2	56	1 - 2	49
3	113	3	113
		PID 3588	iexplore.exe
		WINDOW	CLUSTER
		1 - 2	49
		3	113

#### 4.3.1.3 chrome.exe, iexplore.exe PID = 616, 2976, 2280, 3380, and 3588.

The process flows for chrome.exe and iexplore.exe during time windows of 1 to 3 in Table 4.19 starts in cluster 49 or 56 and ends in cluster 113. All three clusters have port, TCP, and UDP activity. The process flows are behaving as an initialization of the browser and then the processes terminates at window 3. Table 4.20 shows the ports, TCP and UDP features for cluster 49, 56 and 113.

Table 4.20: Hackfest BSOD-19 chrome.exe, iexplore.exe clusters.

cluster	PortsT0	PortsT1	WindowPorts	MINPorts	MAXPorts	AVGPorts	STDPorts	VARPorts
49	0.1765	0	0.3529	0	0.3529	0	0	0
56	0.0694	0	0.3194	0.0278	0.2222	0.0694	0.0278	0.0417
113	0.2131	0	0.2131	0.2131	0.2131	0.2131	0	0
	Q1Ports	MEDPorts	Q3Ports	FFTPorts	FFTPorts2	FFTPorts3	FFTPorts4	FFTPorts5
49	0	0.1765	0	0.3529	0.3529	0.3529	0.3529	0.3529
56	0.0694	0.1181	0.0694	0.3194	0.3046	0.3046	0.2625	0.2625
113	0	0.2131	0	0.2131	0.2131	0.2131	0.2131	0.2131
	TCPT0	TCPT1	WindowTCP	MINTCP	MAXTCP	AVGTCP	STDTCP	VARTCP
49	0.0588	0	0.1176	0	0.1176	0	0	0
56	0.0139	0	0.1389	0	0.0972	0	0	0
113	0.0492	0	0.0492	0.0492	0.0492	0.0492	0	0
	Q1TCP	MEDTCP	Q3TCP	FFTTCP	FFTCP2	FFTCP3	FFTCP4	FFTCP5
49	0	0.0588	0	0.1176	0.1176	0.1176	0.1176	0.1176
56	0.0417	0.0486	0.0417	0.1389	0.1325	0.1325	0.1145	0.1145
113	0	0.0492	0	0.0492	0.0492	0.0492	0.0492	0.0492
	UDPT0	UDPT1	WindowUDP	MINUDP	MAXUDP	AVGUDP	STDUDP	VARUDP
49	0.1176	0	0.2353	0	0.2353	0	0	0
56	0.0556	0	0.1806	0.0278	0.125	0.0556	0.0139	0.0139
113	0.1639	0	0.1639	0.1639	0.1639	0.1639	0	0
	Q1UDP	MEDUDP	Q3UDP	FFTUDP	FFTUDP2	FFTUDP3	FFTUDP4	FFTUDP5
49	0	0.1176	0	0.2353	0.2353	0.2353	0.2353	0.2353
56	0.0278	0.0694	0.0278	0.1806	0.1721	0.1721	0.148	0.148
113	0	0.1639	0	0.1639	0.1639	0.1639	0.1639	0.1639

4.3.1.4 chrome.exe (PID 3676), explore.exe (PID 512). Table 4.21

shows the last two processes that were tracked on BSOD-19 were chrome.exe (PID 3676) and explore.exe(PID 512). These two processes and their process flows did not fit any other patterns of the other process flows on the machines. The behaviors were distinct from the other flows. The start for both processes was cluster 49. That cluster has port, TCP and UDP activity, making it an initialization cluster like the rest of the other browsers. The application explore.exe is not a browser but is the taskbar,

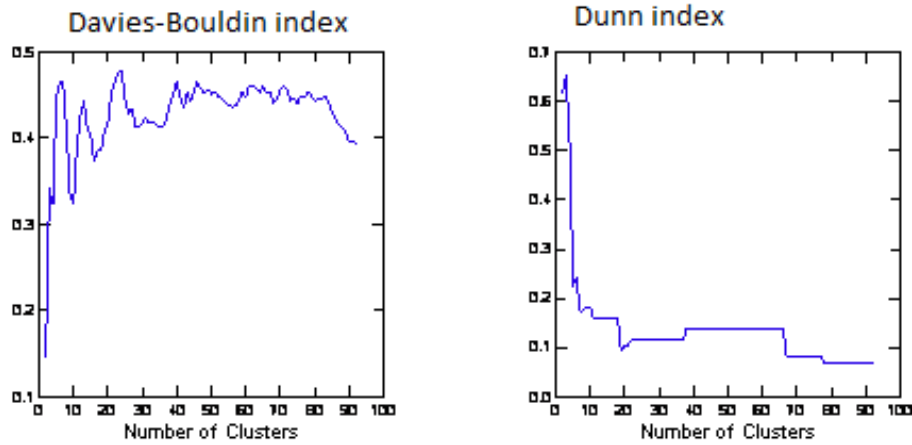
Table 4.21: Hackfest BSOD-19 chrome.exe, iexplore.exe clusters.

PID 3676	chrome.exe	PID 512	explorer.exe
WINDOW	CLUSTER	WINDOW	CLUSTER
1	49	1	49
2 – 14	112	2 – 13	88
15 – 18	33	14	40
19 – 81	112	15 – 18	33
82 – 83	49	19	40
		20 – 78	88
		79 – 83	40
		84	88
		85 – 146	40

desktop and user interface feature. The chrome application starts with cluster 49 and transitions to cluster 112, then to cluster 33, return to cluster 112 and terminates at cluster 49.

The differences between cluster 112 and 33 is that there are more DLLs in cluster 33 than in cluster 112. The amount is roughly 20 more DLLs per discrete time instance. The processes flows associated with explorer.exe show two distinct behavioral clusters, cluster 88 and 40. Looking at the features the key distinct difference was the open registry keys present in cluster 40 and not in cluster 88.

Figure 4.9: Davies-Bouldin Dunn Index Sliding Window 5 on Windows 7



*4.3.2 BSOD - 19 Amount of Clusters needed.* Figures 4.9, 4.10 and 4.11 shows the Dunn and the Davies-Bouldin index as the number of clusters increase for BSOD-19. The graphs were generated in SYSTAT software [42] using the cluster analysis tool for clustering. The algorithm starts by combining clusters into other clusters that are near based on a distance measure, the euclidian distance measure was used. All flows are each individual clusters then iteratively steps through combining cluster with other clusters. A cluster can be represented as a cluster containing only one member. The combining of clusters is performed until all clusters are grouped together. The centroid nearest to one another were combined, distance measurement

Figure 4.10: Davies-Boldin Dunn Index Sliding Window 10 on Windows 7

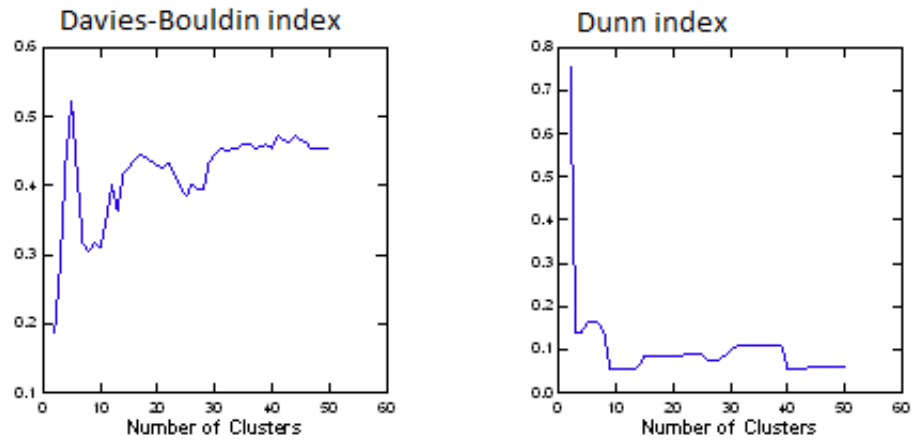
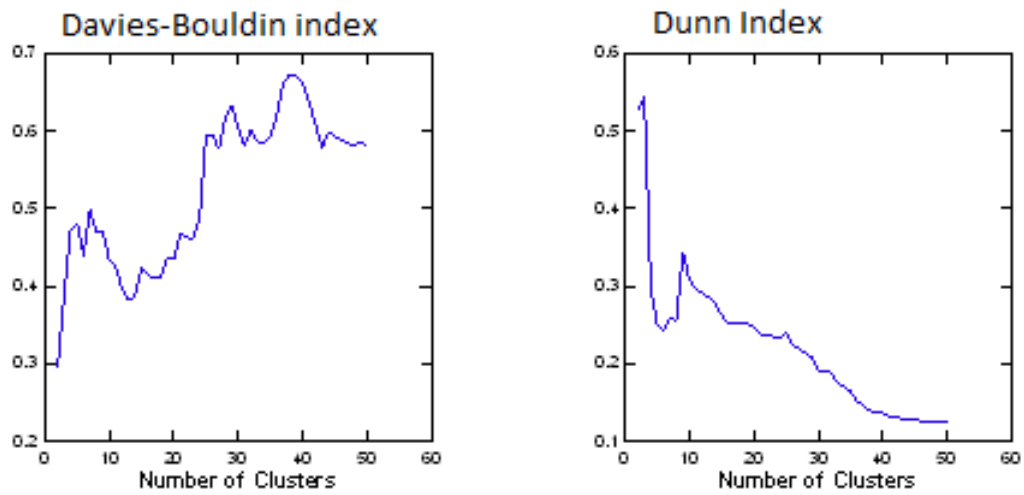


Figure 4.11: Davies-Boldin Dunn Index Sliding Window 20 on Windows 7





against other instances to an established centroid. At points where the number of clusters changes the Dunn and the Davies-Bouldin index was calculated. The graph shows the solution for the cluster were all less than ten. This was not the case because the clusters are too close together and having too small of a cluster value will have some instances closer to clusters not associated to the processes.

Table 4.22: Hackfest CAE-18 TimeLine.

Period	Window	Duration	Frequency $\frac{1}{\text{ematoutput}}$ (sec)	Sensor Capture
1	3 - 66	2011 08 11; 10:34:56 - 2011 08 11; 11:04:38	28	64
2	67 - 113	2011 08 11; 11:12:55 - 2011 08 11; 11:34:32	28	47
3	114 - 223	2011 08 11; 11:42:46 - 2011 08 11; 12:34:35	28	111
4	224 - 245	2011 08 11; 12:42:49 - 2011 08 11; 12:53:49	31	22

Table 4.23: Hackfest CAE-18 chrome.exe, iexplore.exe clusters

PID 552	explorer.exe	PID 576	iexplore.exe	PID 3172	chrome.exe
WINDOW	CLUSTER	WINDOW	CLUSTER	WINDOW	CLUSTER
1 - 241	40	1 - 241	112	1 - 2	21
				3 - 241	85
PID 1804	nessussvrmanag	PID 1984	chrome.exe	PID 3900	iexplore.exe
WINDOW	CLUSTER	WINDOW	CLUSTER	WINDOW	CLUSTER
1 - 2	77	1	79	226 - 227	49
3 - 240	92	2 - 3	101	228 - 241	112
241	98	4 - 240	3		
		241	52		
PID 844	chrome.exe	PID 2772	iexplore.exe	PID 3080	chrome.exe
WINDOW	1	WINDOW	CLUSTER	WINDOW	CLUSTER
1	107	1 - 241	112	1 - 7	112
2 - 95	36			8 - 9	49
96 - 97	100				
98 - 226	36				
227 - 230	3				
231 - 239	36				
240	3				
241	52				

4.3.3 Using BSOD - 19 clusters with other Data Set (CAE-18). Table 4.23 shows the process flows of the new data sets. The evaluation included explorer.exe, iexplore.exe, chrome.exe, and nessvrmanag. The application names were present in both data sets. The dates for CAE-18 was on an earlier date from the training data of BSOD-19 where the clusters was established. The flows were evaluated using the

model of clusters previously established with the data set extracted on BSOD-19. The clusters were previously defined and revaluated using the unseen data set of CAE-18. The behavior of these process flows shows they were being assigned to the predicted flow clusters.

For explorer.exe (PID 552) all flows were assigned to cluster 40. In BSOD-19 the explorer.exe application flows jump from cluster 40 to cluster 88. This shows that the predicted cluster behavior for the new unseen flows was accurate. The explorer.exe was behaving within the behavior of the learned clusters of 40. In BSOD-19 explorer.exe exhibited two behaviors as it had two different cluster assignments through out the life of the process. The feature differences from cluster 88 and 40 are that in cluster 40 registry information was provided as in cluster 88 that set of features were missing. The CAE-18 staying in one cluster indicates that the behavior is the same and that the registry information was present during these times of the calculated features were calculated. A weak point to assigning the registry keys is that the registry key information associated with the clusters are only present during a memory dump of the CMAT sensor. This occurs during the gaps in time and is set for approximately every 30 minutes by default. Looking deeply on the changes of the cluster shows that there is a correlation between the memory dumps when the registry information was included in the flows and when the registry key features were set to zeros when the registry keys information was not available. With this key information shows that the registry information was present in the flows and that the predicted cluster of 40 was the closest cluster and all assigned to that cluster.

The internet browser's process flows in CAE-18 showed identical behavior to each other and similar behavior to the internet browser process flows from BSOD-19. The processes flows of iexplore.exe (PID 576), iexplore.exe (PID 3900), iexplore.exe (PID 2772), and chrome.exe (PID 3080) are the process flows that had similar behavioral clusters . Table 4.25 shows the similarity based on the same cluster assignment of these processes flows. As shown in Table 4.23 the clusters process flows that represents these browsers were landing in cluster 49 and 112. Looking at the features

between the two shows that cluster 49 has ports, TCP, and UDP activity associated with the cluster as 112 does not. Cluster 49 is then the communication cluster and that was previously identified when BSOD-19 was evaluated and learned the cluster behavior association. In Table 4.23 shows that chrome.exe and iexplore.exe setup cluster behavior started with cluster 49. The rest of the process flows landed in cluster 112. BSOD-19 in Table 4.23 shows that cluster 112 was one of the cluster a chrome.exe was assigned to. It also shows that cluster 49 was the first and last cluster the same chrome.exe was assigned to. This gives a good indication that the browsers in the new data sets are behaving like they should be by correlating similar cluster memberships from the training data set of BSOD-19.

The other browsers on CAE-18 were chrome.exe (PID 3172), chrome.exe (1984), and chrome.exe (PID 844). The flows associated with these processes did not have similar behavior to each other but looking at the clusters established using BSOD-19 did fall into the predicted clusters for browsers. For PID 3172 the two clusters process flows were landing in were cluster 21 for the first two time windows and then the flows settles down to cluster 85. Cluster 21 was not in the beginning clusters for the browsers in BSOD-19 but with further investigation the start cluster has ports, TCP and UDP activity within the cluster's behavior. Cluster 21 was not a predefined cluster behavior in BSOD-19 but the characteristics is similar to starting a browser start up with ports, TCP and UDP features associated with the flows. Cluster 85 was a predefined cluster behavior for chrome.exe shown in Table 4.15. For PID 1984 the process flows starts in cluster 79 a predefined initialization browser cluster shown in Table 4.15 for chorme.exe and iexplore.exe. The next two clusters its assigned to was 101 and the startup sequence follows what the nessus application in BSOD-19 startup sequence followed. There might be some relationship to the nessus application running in this machine and the other internet browser chrome.exe (PID 844). As shown in Table 4.23 both has cluster 3 associated with the flows of these processes. The difference of cluster 3 and cluster 36 was that it had more DLLs. This indicates that more activity was going on during those times such as user interaction. The

nessusvrmanag application name with PID 1804 was assigned to cluster 77 for the first two time windows then transitions to cluster 92 for the most of its process life from time window 3 to 24 and terminates in cluster 98 during time window 241. Looking at the past nessus application from BSOD-19 in Table 4.17 shows that cluster 77 was part of the startup setup for nessusvrmanag application and also shows a large portion of the flows are assigned to cluster 92.

Clustering the flows from one machine then evaluating the unseen data set with similar applications running will result into clusters with expected behavior clusters for similar process flows. The application present on both Windows machines were chrome.exe, iexplore.exe, nessessusvrmanag, and explorer.exe. The the new data set was reevaluated using the predefined clusters and assigned to the closest one based on the euclidian distance. The new flows of similar behavior for the unseen process flows in CAE-18 accurately selecting the predicted clusters for application name chrome.exe, iexplore.exe, nessessusvrmanag, and explorer.exe.

*4.3.4 Data Collect and Problems.* The problem with the uncontrolled experiments was that few application were running. The processes found in memory were predominantly native operating system processes like svchost.exe, lsass.exe and others that were difficult model. As a result of the lack of processes running on both machines the analysis were limited to the four processes chrome.exe, iexplore.exe, nessessussvrmanag, and explorer.exe. Even with promising results of these four processes a gold standard of process flows with labels of normal and anomalous still needs to be developed to give a definite accuracy for the clusters. This research is just the first step and lays the ground work for process flows and how they may be used in the computer security world.

The number of native windows application were present in many clusters and there was no way to narrow them down to on cluster or a few clusters. The behavior is too dynamic where its responsible and can do many things in the system. A process like svchost.exe displayed these characteristics. A flaw in using the clusters are a

malicious software might identify themselves as `svchost.exe` and then can get lucky and go undetected because it will be assigned to cluster that `svchost.exe` is present because in the training data its assigned to many clusters.

If a processes has a PID and in CMAT it identifies itself as one of these processes, the flow generated would find the nearest centroid. The processes would be labeled normal if it was one within the cluster's threshold. The processes like `svchost.exe` and `smss.exe` where the majority of the observable processes in CMAT was made it difficult for labeling clusters based on application types when application were located in many clusters across the solution space. The dissimilarity of each instance of `svchost.exe` made it belong to several clusters in the developed model. A malware just needs to change its application name it provides to the computer system, then under this the malware would go undetected.

#### **4.4 *Summary***

This chapter provides some interesting observations of the resulting clusters of process flows. Solving the intrusion problem using a novel heuristic approach takes time and numerous iterations of improvement. Providing this novel approach to the research community this method could be improved or totally changed using different machine learning techniques. Anomaly detection was always plagued with high false positives alerts. This research does not tackle reducing false positive alerts for systems that have already been developed but to propose a whole new approach all together. An exploratory approach was adopted where this new idea on how to think of processes in memory are characterized and using this new thinking determine if it can be used somehow for detecting the intrusion on a computer. The flaws limited a thorough verification test on the system. The data collection had flaws associated with data collection tool malfunctions. The goal was to demonstrate on the data that was considered valid that clustering processes flows with similar behavior was possible with the novel way to describe a process. With improvements and maturity of the sensor this approach can be examined in the future. The exploratory results presented

here shows promising results that future research based on this idea of “Host-Based Process Flow Features” is a worth while direction to investigate for a new anomaly based IDS method.

## V. Conclusions

The results of this thesis demonstrates that processes described as dynamic statistical behavior in memory can result in a usable structure for identifying behavior. Activities on a computer for two data sets were generated to determine if host derived flow based features could be used to behaviorally identify the activity. The first data set consisted of scripted events in which all of the user activities and the timing of those activities are known. The second makes use of an unknown situation and is a collection of data from the HACKFEST exercise. Unfortunately, both of these data collections have flaws associated with data collection tool malfunctions. The flaws appear as gaps in the timeliness of the collection, increasing delays, and missing data. The analysis presented in this thesis is primarily exploratory and is an attempt to inspire future research in the use of “Host Based Process Flow Features.” The analysis does show that the flow features do result on a strong behavioral clustering. However, because of the problems with the data no definitive conclusions are drawn.

Assuming no inconsistency with the data, this method of clustering did converge. An open issue is to identify a systematic way to come up with the threshold value. Only one type of anomalous process was created and only tested against notepad.exe, with the threshold value working as the maximum distance a cluster has as part of its members. This could produce lots of error because some clusters are densely populated only containing one type of application. This makes the threshold really small and needs to be adjusted as new processes are assigned to that cluster and identified as not anomalous.

The data collect was flawed with missing time slices or unusually large gaps in time. This greatly affected the centers of the flows. The generation of the flows was an inefficient implementation resource wise. The intent was to determine if statistical process flow was attainable. Progress was made on the steps to generate flows but needs to be more efficient. The sensor created thousands of files making associations of statistical process behavior to flows difficult and timely.

### ***5.1 Limitations and Assumptions***

The biggest limitation was dealing with real world data. The generation of flows was time consuming because of the amount of data and dealing with inconsistent time events of sensor outputs. All of these limitation can drive to a wrong solution. To deal with this time intervals inconsistency less than one minute of the CMAT feature files was considered the same where the sliding window treated the five equally spaced time events as a flow. The gaps where it was greater than a couple of minutes was treated as a stopping criteria for the sliding window. The sliding window would start after the gap in time. The result of doing this does shift the flows to other cluster because the statistical changes occur especially if the gap in time were measured in the hours.

In all anomaly detection methods the assumption is anything outside the norm of a baseline characteristics in this case a cluster with a threshold is anomalous. This might not be the case as various processes are known to be too diversified in feature space where a processes like a an internet browsers would spawn many new processes running to help handle the browser. Windows process like svchost.exe was the do all things process for the operating system. This was problematic coming up with a pattern to where these types of flow will be assigned in the cluster space. A browser might have flash, visiting some website this will introduce new processes associated with the browser. The browser would create another processes that handles the flash management. New research needs to enhance this method as to include linking other process that it spawns. A parent and child process relationship feature as part of the 194 vector of process flow should be included.

This research narrowed the observation to a few application types because not all processes can be contained in a few cluster. The lack of user inputs to processes might have affected the clusters of an application type to jump to other clusters, but because of the lack of user interaction the few applications being tracked tend to stay in one cluster during the whole time of the data collect.



*5.1.1 Sensor Impact and Virtual Environment.* The system is operating in a virtual machine environment will have distinct differences on performances because its capturing the guest's OS memory. The performance differences and models created using this method may affect the accuracy. Depending on the hosts machine where the CMAT sensor is placed and running it might have slight difference on how the flows are calculated. The sensor was assumed to have the same behavior without slowing the the host down. During the the data capture it was evident that the system was slowing down the guest operating system because of the lack of responsiveness by user input in mouse movement, and keyboard inputs.

*5.1.2 Memory as an Observable Environment.* The only way to have 100% observable of the threat space is to have every processes in a computer captured for every permutations the processes can make. Then for each process have the corresponding flows for each permutation. Also have the correct distribution of permutation for each processes. After having all the flows for each permutation for each processes then a clustering algorithm can provide the optimal solution.

Even if this was possible a signature method with an updating database of known malicious behavior would be more feasible. A local solution is only possible and a partial view of the threat space. To maximize this method and capture most of the processes behavior a good data collect processes must be implemented. The data collect ideally should be long enough that describes the normal behavior of each processes of interest. The processes that spawn other processes was not handled and linked to parent processes as there was no way to link this information from the CMAT feature files.

## **5.2 Contributions**

The contributions to the field of computer security is an alternative approach to view processes in memory. This alternative view can aid to possible detection of anomalous processes. This research provides the feature set of dynamic statistical

behavior of processes in memory called process flow. Processes are the active entity of programs and by tapping into this data source a method was proposed to capture a behavior of a processes from a starting time to an end time in memory. This novel approach is limited to the processes running on the target system but could be implemented on other machines. The only roadblock is the current sensor works only in virtual environment setting. If the sensor could work on an operating system without a virtual machine and have the sensor protected against attack this method could be ported to other system like a computer system with the standard configuration of the United States Air Force or supervisory control and data acquisition (SCADA) computers. The assumptions of SCADA computers are they have limited processes. The threat space and clusters would be much less than a home personal computer.

### ***5.3 Future Work***

The future work must include capturing more data with diversified processes running. At the time of the data collect the realization of user interaction and making processes iterate through permutations of different behavior was clearly lacking. A target system of different types could also be for future focus. The USAF has a standard configuration for a user computer. Creating models for these computer system might be worthwhile for protecting the USAF working environment.

The feature set could be updated and studied on the possible redundancy some features have describing the process flow. Reducing feature set could help guide to a solution faster doing unnecessary calculations. There are many points of failure to this method but the one that stands out is the flow generation. To have good flows providing consistent data the sensor must be reliable. Part of the reason on inconsistent data collect was due to user error and have wrong configuration for the sensor.

The increase in tablets and cell phones as a computing platform are increasing in popularity. Attackers might hone on these systems knowing users performs tasks like banking, checking email, and possible shopping. The increase in use such as doing

banking transaction on these mobile devices increases the threat of identity theft and stealing banking information. Future work could take a look at programs in these systems and apply the processes flows to these machines. The future work are endless but for any proposed heuristic approach to solving detection intrusions the need to decrease false positive indication is the primary reason why these systems are not found in most commercial markets. This research does not propose any replacement of an IDS but just proposes an alternative view of the way a processes is viewed in memory. New research can use this as the foundation of process flow and apply more innovative approach to detecting intrusions.

## *Appendix A. Normal Activity Script*

### ***A.1 Script Labeled Data Collect***

Table A.1: The normal activity script for computer  
 Start Computer Name Description

11:38	7.5	Opens Personal Information File in share drive. Open Payroll document. Excel document
11:43	7.5	Opens Internet Explorer. Googles Dogs Pictures.
11:46	7.5	Downloads Doggy Picture From Internet View Doggy Picture on Windows Photo Viewer
11:48	7.5	Read Email, Microsoft Outlook
11:51	7.5	Close Outlook, Excell, and photo viewer
11:53		Googled "Divorce in Ohio"
11:55	7.4 , 7.5	TIME CHANGED ONE HR AHEAD
12:57	7.5	Down Divorce Documents for Court PDF file "otf.pdf"
12:59	7.4	Google Search "Cage fighting"
13:03	7.5	Opened Calc.exe
13:03	7.4	video results in search need flash installed to watch video. Installed flash from adobe.com and installed flash
13:06	7.5	Opened sticky notes software
13:10	CMAT	Memory Capture Seg Fault, Had to restart
13:12	7.4	Logged Off, 1 min later Logged on

13:14	7_4	Cage Fighting google search
13:15	7_4	Google Search "Octagon"
13:17	7_4	Google search "Cage Fighting Schedule dc"
13:20	7_5	Google search "Divorce Law"
13:20		Seg Fault Restarted CMAT
13:27	7_5	Opened outlook email
13:29	7_4	Opened outlook
13:32	7_4	Sent Email to Scarlet and Peacock
14:54	7_4	Google Search "Cage fighting schedule dc"
14:59	7_4	Downloaded Google chrome. IE keeps crashing
15:01	7_4	Started Using Chrome. Google Search "Cage fighting women dc"
15:04	7_4	Youtube Search "Cage Fighting Women"
15:05	7_4	Google Search "Cock Fighting"
15:05	7_4	Google Search "Cock Fighting DC"
15:08	7_4	Google Search "Cock Fighting Schedule"
15:15	7_4	Email Reply To Scarlet

15:18	7.5	Opened Paint. Made a Paint document and altered puppy picture that was saved on desktop
15:20	7.5	Opened Calculator
15:21	7.5	Played Kalimba music (Under Sample Music), Could not play because not sound installed
15:23	7.5	Opened MS Excel
15:24	7.5	Opened MS power Point
15:24	7.5	Opened CMD window
15:25	7.5	Opened MS Words
15:27	7.5	dir command on CMD window
15:28	7.5	saved MS Excel named "numtest"
15:34	7.4	Sent Email response to Mr Green
15:36	7.5	Downloaded Chrome and installed chrome
15:38	7.5	Downloaded Firefox and installed firefox

15:41	7.5	opened firefox, chrome, and firefox
15:42	7.5	Using firefox went to ESPN.com
15:43	7.5	Using chrome went to facebook.com
15:45	7.5	Using IE went to reddit.com
16:03	7.4	Network Drive Opened "Joint Strike Fighter Program" MS WORD and printed the document. Printed it twice
16:14	7.4	Went to Pandora website and listened to internet radio. "Britney Spears Radio Stationed". Site not letting music play.
16:21	7.5	Open remote desktop shell to ms01
16:25	7.5	Opend classified project picture from remote desktop
16:27	7.4	Copied "Sleep Away" from Sample Music to Desktop using CTR C CTRV command
16:29	7.4	Copied "Sleep Away" from desktop to Document Library using CTR C CTR V
16:42	7.4	Created shortcut of shared network to desktop
16:43	7.4	Copied File "JointStrikeFighter" Work document from shared folder "Projects" to Desktop, Using Drag and Drop

16:48	7_4	Went to YouTube using Chrome
16:49	7_4	Went to facebook using another tab in Chrome
16:50	7_4	Went to Gmail.com using another tab in chrome
16:50	7_4	Went to reddit.com using another tab in chrome
16:52	7_4	Opened Notepad
16:54	7_4	Opened Calc.exe
16:56	7_5	Logged from remote shell
16:57	7_5	Went to facebook.com using another tab in IE
16:58	7_5	Went to Netflix.com using another tab in IE
16:59	7_5	Went to HULU using another tab in IE
17:04	7_5	Email Response Sent to Scarlet
17:07	7_4	Email Response to MR GREEN
17:10	7_5	Went to ESPN.COM using another tab in IE
17:54	7_4	Created Word Doc for Fighter Design proposal in Documents Folder
17:55	7_4	Copy "draft copy of fighter design" from Document Folder to Desktop
18:05	7_4	cut and paste some text from "JointStrike-Fighter" Word Document on desktop to "draft copy of fighter design" Word document on desktop using right click mouse. Saved Document
18:10	7_4	Copy "draft copy of fighter design" from desktop to network folder "Projects" using mouse cut and paste
18:12	7_5	closed all tabs of ID running
18:13	7_5	Closed all windows
18:24	7_4	Created folder in C:\called "White_Folder"
18:25	7_4	Copied and Pasted 2 files using Highlight and right click copy and paste. Desktop to another folder in "C:\White_Folder"
18:28	7_5	Down loaded pdf.reader and installed it
18:37	7_5	Opened Divorce Document on Desktop and started filling it out
18:38	7_5	Printed the divorce document



## Bibliography

1. M. Christodorescu, S. Jha, S. Seshia, D. Song, and R. Bryant, "Semantics-aware malware detection," in *Security and Privacy, 2005 IEEE Symposium on*, pp. 32–46, IEEE, 2005.
2. P. Garcia-Teodoro, J. Diaz-Verdejo, G. Macia-Fernandez, and E. Vazquez, "Anomaly-based network intrusion detection: Techniques, systems and challenges," *computers & security*, vol. 28, no. 1-2, pp. 18–28, 2009.
3. I. Mukhopadhyay, M. Chakraborty, and S. Chakrabarti, "A comparative study of related technologies of intrusion detection & prevention systems," *Journal of Information Security*, vol. 2, pp. 28–38, 2011.
4. T. Shon and J. Moon, "A hybrid machine learning approach to network anomaly detection," *Information Sciences*, vol. 177, no. 18, pp. 3799–3821, 2007.
5. M. Laureano, C. Maziero, and E. Jamhour, "Protecting host-based intrusion detectors through virtual machines," *Computer Networks*, vol. 51, no. 5, pp. 1275–1283, 2007.
6. D. Malan and M. Smith, "Exploiting temporal consistency to reduce false positives in host-based, collaborative detection of worms," in *Proceedings of the 4th ACM workshop on Recurring malware*, pp. 25–32, ACM, 2006.
7. C. Feng, J. Peng, H. Qiao, and J. Rozenblit, "Alert fusion for a computer host based intrusion detection system," in *Engineering of Computer-Based Systems, 2007. ECBS'07. 14th Annual IEEE International Conference and Workshops on the*, pp. 433–440, IEEE, 2007.
8. G. Chilton, "Cyberspace leadership: Towards new culture, conduct, and capabilities," *Air and Space Power Journal*, vol. 23, no. 3, pp. 5–10, 2009.
9. G. Vigna and C. Kruegel, "Host-based intrusion detection," *Handbook of Information Security. John Wiley and Sons*, 2005.
10. R. Bace, "Nist special publication on intrusion detection systems," tech. rep., DTIC Document, 2001.
11. Y. Chen, Y. Li, X. Cheng, and L. Guo, "Survey and taxonomy of feature selection algorithms in intrusion detection system," in *Information Security and Cryptology*, pp. 153–167, Springer, 2006.
12. T. Daniels and E. Spafford, "Identification of host audit data to detect attacks on low-level ip vulnerabilities," *Journal of Computer Security*, vol. 7, no. 1, pp. 3–35, 1999.
13. R. Kemmerer and G. Vigna, "Intrusion detection: a brief history and overview," *Computer*, vol. 35, no. 4, pp. 27–30, 2002.

14. G. Tandon and P. Chan, "On the learning of system call attributes for host-based anomaly detection," *International Journal of Artificial Intelligence Tools*, vol. 15, no. 6, pp. 875–892, 2006.
15. W. Lee and S. Stolfo, "Data mining approaches for intrusion detection," in *Proceedings of the 7th conference on USENIX Security Symposium-Volume 7*, pp. 6–6, Usenix Association, 1998.
16. T. Daniels and E. Spafford, "Identification of host audit data to detect attacks on low-level ip vulnerabilities," *Journal of Computer Security*, vol. 7, no. 1, pp. 3–35, 1999.
17. F. Maggi, M. Matteucci, and S. Zanero, "Detecting intrusions through system call sequence and argument analysis," *IEEE Transactions on Dependable and Secure Computing*, pp. 381–395, 2010.
18. R. Moskovitch, S. Pluderman, I. Gus, D. Stopel, C. Feher, Y. Parmet, Y. Shahr, and Y. Elovici, "Host based intrusion detection using machine learning," in *Intelligence and Security Informatics, 2007 IEEE*, pp. 107–114, IEEE, 2007.
19. E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo, "A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data," *Applications of Data Mining in Computer Security*, vol. 6, pp. 77–102, 2002.
20. J. Erskine, G. Peterson, B. Mullins, and M. Grimaila, "Developing cyberspace data understanding: using crisp-dm for host-based ids feature mining," in *Proceedings of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research*, p. 74, ACM, 2010.
21. J. Ji, "Holistic network defense: Fusing host and network features for attack classification," tech. rep., DTIC Document, 2011.
22. H. Carvey, *Windows Forensics and Incident Recovery (The Addison-Wesley Microsoft Technology Series)*. Addison-Wesley Professional, 2004.
23. X. Hoang, J. Hu, and P. Bertok, "A program-based anomaly intrusion detection scheme using multiple detection engines and fuzzy inference," *Journal of Network and Computer Applications*, vol. 32, no. 6, pp. 1219–1228, 2009.
24. S. James and J. Nordby, *Forensic science: an introduction to scientific and investigative techniques*. CRC, 2005.
25. I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *The Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, 2003.
26. S. Zanero and S. Savaresi, "Unsupervised learning techniques for an intrusion detection system," in *Proceedings of the 2004 ACM symposium on Applied computing*, pp. 412–419, ACM, 2004.

27. L. Portnoy, E. Eskin, and S. Stolfo, "Intrusion detection with unlabeled data using clustering," in *In Proceedings of ACM CSS Workshop on Data Mining Applied to Security (DMSA-2001)*, Citeseer, 2001.
28. W. Wang, X. Guan, X. Zhang, and L. Yang, "Profiling program behavior for anomaly intrusion detection based on the transition and frequency property of computer audit data," *computers & security*, vol. 25, no. 7, pp. 539–550, 2006.
29. T. Shon and J. Moon, "A hybrid machine learning approach to network anomaly detection," *Information Sciences*, vol. 177, no. 18, pp. 3799–3821, 2007.
30. A. Sperotto, G. Schaffrath, R. Sadre, C. Morariu, A. Pras, and B. Stiller, "An overview of ip flow-based intrusion detection," *Communications Surveys & Tutorials, IEEE*, vol. 12, no. 3, pp. 343–356, 2010.
31. C. Zhou, C. Leckie, and S. Karunasekera, "A survey of coordinated attacks and collaborative intrusion detection," *Computers & Security*, vol. 29, no. 1, pp. 124–140, 2010.
32. D. Yeung and C. Chow, "Parzen-window network intrusion detectors," in *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, vol. 4, pp. 385–388, IEEE, 2002.
33. Y. Li, B. Fang, L. Guo, and Y. Chen, "Network anomaly detection based on tcm-knn algorithm," in *Proceedings of the 2nd ACM symposium on Information, computer and communications security*, pp. 13–19, ACM, 2007.
34. A. Moore, D. Zuev, and M. Crogan, "Discriminators for use in flow-based classification," *Technical report, University of Cambridge, Computer Laboratory*, 2005.
35. M. Roughan, S. Sen, O. Spatscheck, and N. Duffield, "Class-of-service mapping for qos: a statistical signature-based approach to ip traffic classification," in *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pp. 135–148, ACM, 2004.
36. J. Okolica and G. Peterson, "A compiled memory analysis tool," *Advances in Digital Forensics VI*, pp. 195–204, 2010.
37. W. Li, A. Canine, M. Moore, and R. Boola, "Efficient application identification and the temporal and spatial stability of classification schema," *Computer Networks*, vol. 53.
38. I. Witten, E. Frank, and M. Hall, *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2011.
39. I. Witten, E. Frank, and M. Hall, *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2011.
40. D. Davies and D. Bouldin, "A cluster separation measure," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, no. 2, pp. 224 – 227, 1979.

41. J. Bezdek and N. Pal, "Cluster validation with generalized dunn's indices," in *Artificial Neural Networks and Expert System, 1995. Proceedings., Second New Zealand International Two-Stream Conference on*, pp. 190 – 193, IEEE, 1995.
42. L. Wilkinson, G. Blank, and C. Gruber, *Desktop Data Analysis SYSTAT*. Prentice Hall PTR, 1996.

<b>REPORT DOCUMENTATION PAGE</b>					<i>Form Approved</i> <b>OMB No. 0704-0188</b>	
The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. <b>PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.</b>						
<b>1. REPORT DATE</b> (DD-MM-YYYY) 14-06-2012		<b>2. REPORT TYPE</b> Master's Thesis		<b>3. DATES COVERED</b> (From — To) Sep 2010 — Jun 2012		
<b>4. TITLE AND SUBTITLE</b>  <div style="text-align: center;">Process Flow Features as a Host-Based Event Knowledge Representation</div>				<b>5a. CONTRACT NUMBER</b>		
				<b>5b. GRANT NUMBER</b>		
				<b>5c. PROGRAM ELEMENT NUMBER</b>		
<b>6. AUTHOR(S)</b>  Pacer, Benhur, E., Captain, USAF				<b>5d. PROJECT NUMBER</b>		
				<b>5e. TASK NUMBER</b>		
				<b>5f. WORK UNIT NUMBER</b>		
<b>7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)</b> Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 Hobson Way WPAFB OH 45433-7765				<b>8. PERFORMING ORGANIZATION REPORT NUMBER</b>  AFIT/GCS/ENG/12-06		
<b>9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)</b>  intentionally left blank				<b>10. SPONSOR/MONITOR'S ACRONYM(S)</b>		
				<b>11. SPONSOR/MONITOR'S REPORT NUMBER(S)</b>		
<b>12. DISTRIBUTION / AVAILABILITY STATEMENT</b>  Approval for public release; distribution is unlimited.						
<b>13. SUPPLEMENTARY NOTES</b>  This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.						
<b>14. ABSTRACT</b> The detection of malware is of great importance but even non-malicious software can be used for malicious purposes. Monitoring processes and their associated information can characterize normal behavior and help identify malicious processes or malicious use of normal process by measuring deviations from the learned baseline. This exploratory research describes a novel host feature generation process that calculates statistics of an executing process during a window of time called a process flow. Process flows are calculated from key process data structures extracted from computer memory using virtual machine introspection. Each flow cluster generated using <i>k</i> -means of the flow features represents a behavior where the members of the cluster all exhibit similar behavior. Testing explores associations between behavior and process flows that in the future may be useful for detecting unauthorized behavior or behavioral trends on a host. Analysis of two data collections demonstrate that this novel way of thinking of process behavior as process flows can produce baseline models in the form of clusters that do represent specific behaviors.						
<b>15. SUBJECT TERMS</b>  Host-Based, Intrusion Detection System, Process Flow Features, <i>k</i> -means clustering						
<b>16. SECURITY CLASSIFICATION OF:</b>			<b>17. LIMITATION OF ABSTRACT</b>	<b>18. NUMBER OF PAGES</b>	<b>19a. NAME OF RESPONSIBLE PERSON</b>	
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. Gilbert L. Peterson	
U	U	U	UU	117	<b>19b. TELEPHONE NUMBER</b> (include area code) (937) 785-6565; Gilbert.Peterson@afit.edu	